il NANOBOOK® Z-80™

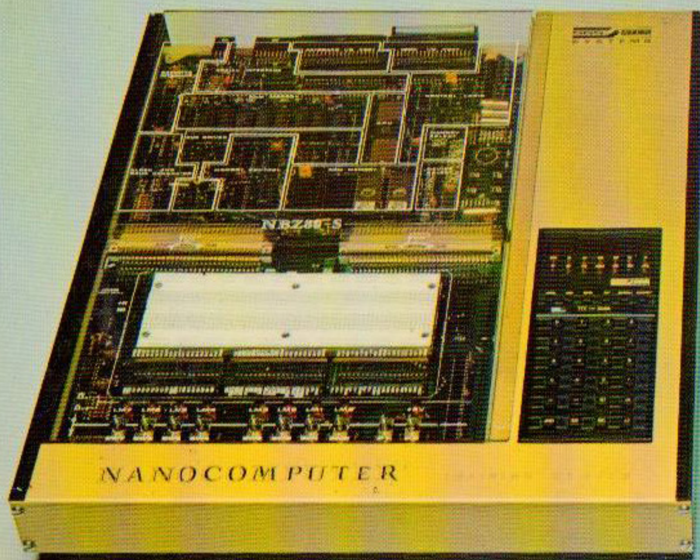
VOL. 3 - TECNICHE DI INTERFACCIAMENTO

EDIZIONE ITALIANA

E. A. NICHOLS
J. C. NICHOLS
P. R. RONY

JACKSON ITALIANA EDITRICE





il NANOBOOK® Z-80™

VOL. 3 - TECNICHE DI INTERFACCIAMENTO

di
Elizabeth A. Nichols, Joseph C. Nichols,
e Peter R. Rony



GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12 - 20124 Milano

Prima edizione: 1980

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

PREFAZIONE AI NANOBOOK®

I dispositivi elettronici integrati su grande scala hanno determinato una rivoluzione nella concezione degli apparati elettronici, rendendo disponibile a basso costo la flessibilità e la potenza operativa dei calcolatori elettronici.

I calcolatori elettronici, presenti negli anni 50 solo in specializzati centri di calcolo o in centri organizzativi e amministrativi di grandi Enti, sono passati negli anni 60, spesso col nome di minicalcolatori, nei laboratori scientifici e nelle unità produttive delle industrie, e sono oggi presenti nella forma di microcalcolatori sul tavolo di ogni tecnico, nelle tasche degli studenti, all'interno di prodotti industriali di uso comune, automobili, bilance, registratori di cassa, giochi, ecc.

Questa diffusione, che oggi è esplosiva, determina la necessità di una corrispondente esplosiva diffusione nei tecnici della conoscenza dei meccanismi di funzionamento dei calcolatori elettronici, dei loro linguaggi, delle tecniche di interfacciamento tra essi e gli organi da cui devono prendere informazioni o gli organi che essi devono comandare o che sono destinati a presentare l'informazione da essi elaborata.

La conoscenza degli aspetti strumentali (hardware) e logicali (software) del mondo dei calcolatori è divenuta strumento di lavoro non solo di specialisti ma di ogni ingegnere elettronico e di ogni tecnico a livello intermedio.

I grandi e medi calcolatori potevano essere conosciuti dagli utilizzatori in modo molto mediato, perchè si presentavano "vestiti" con linguaggi evoluti, "Fortran , Cobol, Basic", assai vicini ai linguaggi naturali degli operatori umani; i microcomputer, pur potendosi presentare, nelle forme più evolute, "vestiti", sono molto interessanti quando sono "nudi" e devono essere conosciuti nei loro linguaggi di macchina: infatti nella massima parte dei casi i microcomputer, in sè versatilissimi, sono destinati non a divenire dei calcolatori "universali" ma dei calcolatori "dedicati", destinati cioè a un lavoro particolare abbastanza semplice ma da effettuare con la massima efficienza e rapidità: le compilazioni o interpretazioni di linguaggi evoluti, con i relativi tempi di attesa e la minore efficienza dei programmi oggetto così ottenuti, non sono in genere sopportabili.

Bisogna "parlare" al microcomputer col suo linguaggio. Questa necessità è anche un'opportunità per i tecnici, a dir il vero con fatica, di conoscere bene e da vicino i calcolatori.

Ho detto con fatica e non con difficoltà soprattutto presentando questi libri che trattano degli "scarafaggi neri": questi meravigliosi oggetti della tecnica della grande integrazione, dal corpo nero e dai molti piedini, che bisogna imparare ad usare ma che solo pochi specialisti devono saper costruire.

Gli "scarafaggi neri" si imparano ad usare dall'esterno per le loro proprietà funzionali, così come i cavalieri imparano ad usare il cavallo senza pretendere di saperlo costruire, ma quanta arte per cavalcare facendo magari le evoluzioni di squadra!

Questi libri insegnano ad usare i microcalcolatori nelle loro unità funzionali: unità centrale, memoria di accesso casuale, memoria di sola lettura, interfaccia di ingresso

uscita, senza richiedere al lettore quasi nessuna conoscenza tecnica specialistica, ma presupponendo in lui, semplicemente, capacità logiche.

E' un approccio pragmatistico e sperimentale di originale concezione didattica: chi li legge, eseguendo diligentemente sull'associato Nanocomputer gli esercizi, impara a conoscere le tecniche di programmazione e di interfacciamento dei microcomputer e i loro linguaggi di macchina e mnemonici (in particolare quelli dello Z-80) e il potente corredo d'istruzioni che fa di questi strumenti degli oggetti meravigliosi.

Il novellino è un po' impressionato dal fatto che una macchina così potente come è il suo Nanocomputer richieda tanta programmazione e studio per fare una moltiplicazione in virgola mobile, mentre il suo calcolatorino tascabile Texas o Hewlett Packard fa operazioni e sequenze di operazioni molto più complicate premendo solo pochi tasti, ma va subito avvertito che il nanocomputer è una macchina "nuda" e se, una volta conosciutala egli la vuole vestire, potrà con molto lavoro di logicale (software) emulare e superare le capacità del suo calcolatore tascabile.

Benvenuti dunque questi libri, e quelli che seguiranno, che apriranno la possibilità a chiunque sia dotato di buona volontà di accedere a partecipare attivamente alla costruzione delle macchine "intelligenti" (ma rispetto all'uomo sempre stupide) del futuro.

Prof. Emilio Gatti
Direttore dell'Istituto di Fisica
del Politecnico di Milano

Milano, 1980

INTRODUZIONE DEGLI AUTORI

Stiamo assistendo ad una vera e propria rivoluzione nel campo della microelettronica, sempre più impetuosa.

Tutto cominciò trent'anni fa, con lo sviluppo del transistore.

Il transistore, piccolo amplificatore a basso consumo, rimpiazzò la vecchia valvola ad alto assorbimento di potenza, utilizzata dai computer della prima generazione.

Grazie al sinergismo tra i transistori e la logica digitale, le dimensioni ridotte ed il basso costo, questi dispositivi sono divenuti gli elementi costruttivi di base della circuiteria di un computer. I transistori si combinano per formare porte logiche; con queste si possono costruire flip-flop, contatori, sommatori e altre funzioni logiche; questi ultimi, a loro volta, possono essere utilizzati per realizzare memorie, funzioni di controllo e unità aritmetico-logiche (ALU) che, finalmente, sono necessarie per costruire le unità centrali di processo (CPU) di un computer.

Così, il numero di transistori in un circuito logico è misurabile in rapporto alla complessità delle funzioni svolte. Nel 1959 vennero sviluppati su sottili fette di silicio o germanio, i primi circuiti integrati, che consistevano in piccoli gruppi di transistori planari. Con questi, l'era dell'Integrazione a Piccola Scala (SSI) era incominciata: era possibile integrare fino a 12 porte su un solo circuito integrato. Dal '59 in poi, il numero di transistori contenuti nei circuiti integrati più avanzati, è praticamente raddoppiato ogni anno.

Oggi sono disponibili circuiti con 262.144 elementi e l'evoluzione tecnologica è ancora lontata dai limiti teorici.

La CPU Z-80 ed i chip di supporto, introdotti sul mercato dalla Zilog nel 1976, rappresentano lo stato dell'arte nel campo dei microprocessori a 8 bit.

La Zilog ha ora realizzato il successore dello Z-80: la CPU e i chip di supporto della serie Z-8000. Quest'ultima, tuttavia, è una CPU con un'intelligenza paragonabile ad un minicomputer di media capacità: un salto tecnologico significativo. La SGS-ATES, già nota come sorgente alternativa della serie Z-80, sta anch'essa per iniziare lo sviluppo, con tecnologia propria, della serie Z-8000.

Nonostante tutti i progressi fin qui delineati, siamo soltanto all'inizio. Ci si potrà rendere conto realmente di questa rivoluzione, osservando la proliferazione con andamento esponenziale di prodotti e servizi basati sulla microelettronica.

Questo libro è il terzo di una serie di volumi sulle tecniche di programmazione e di interfacciamento del microprocessore Z-80.

Il primo volume è dedicato al software dello Z-80: in particolare alla programmazione in linguaggio macchina ed in linguaggio assembler.

Il secondo volume è invece dedicato all'elettronica digitale, mentre il terzo tratta i problemi di interfacciamento con gli elementi CPU, PIO e CTC della famiglia Z-80.

Questi libri sono strutturati come testi-laboratorio e sono pensati in modo da fornire un approccio d'insieme sulla programmazione e sull'interfacciamento di un microcomputer.

Ciò che viene posto in risalto è, in primo luogo, il metodo di apprendere attraverso una continua sperimentazione. Ogni argomento introdotto è approfondito attraverso un lavoro di laboratorio, che permette di verificare non solo l'esattezza o meno degli esercizi svolti, ma anche di evidenziare eventuali errori.

I primi due volumi non richiedono alcuna preparazione specifica su computer, tecniche di programmazione ed elettronica digitale. Il terzo presuppone invece una familiarità con gli argomenti trattati nei due precedenti.

In tutti e tre i libri la materia viene presentata in un ordine ritenuto dagli autori il più proficuo per l'apprendimento.

Per ogni esercizio è fornita la risposta e ci si è sforzati di prevenire i dubbi derivati dagli esperimenti, individuando ove possibile, l'eventuale estensione logica dei medesimi.

Per rafforzare questo orientamento "di laboratorio" su cui sono costruiti questi libri, abbiamo ritenuto opportuno utilizzare un microcomputer signle-board molto sofisticato, basato sulla CPU Z-80, realizzato dalla SGS-ATES: il NANOCOMPUTER®. Esso è un eccellente sistema didattico, poichè, pur semplice ad usarsi per un neofita, è tuttavia dotato di sufficienti requisiti di flessibilità, espandibilità e sofisticazione per tenere vivo anche l'interesse dell'utilizzatore più esperto.

Gli autori sono fortemente riconoscenti a molte persone della SGS-ATES di Agrate Brianza - Milano: R. Baldoni, A. Cattania, D. Comboni, B. Facchi, F. Luraschi, P. Madaschi, C.E. Ottaviani, C. Wallace e soprattutto A. Watts, le cui idee e la cui esperienza sul NANOCOMPUTER ®, hanno fortemente arricchito questi libri.

Vorremmo inoltre ringraziare C. Edson e U. Broggi della SGS-ATES in USA, che molto hanno contribuito a facilitare questo progetto, agendo da tramite tra gli USA e l'Italia.

Molto credito è infine dovuto a J. Titus e D. Larsen del Blacksburg Group ed al Dott. Fontana della Microlem Divisione Didattica, per i loro sforzi di coordinamento rispettivamente con la Howard W. SAMS and Co. Inc., per l'edizione americana e con la Jackson Italiana Editrice, per l'edizione italiana; all'Ing. A. Cavalcoli, per la traduzione italiana di questo libro.

Elizabeth A. Nichols
Joseph C. Nichols
Peter R. Rony

SOMMARIO

CAPITOLO 1

INTERFACCIAMENTO DELLO Z80

INTRODUZIONE

In questo capitolo vengono illustrati alcuni degli obiettivi dell'interfacciamento dei microprocessori, vengono date le definizioni di alcuni concetti ad esso relativi e vengono indicate alcune delle metodologie impiegate.

OBIETTIVI

Alla fine di questo capitolo sarete in grado di:

- Spiegare che cosa vuol dire interfacciare un microprocessore e fare degli esempi di applicazioni riguardanti l'interfacciamento stesso.
- Definire gli obiettivi dell'interfacciamento dei microprocessori.
- Descrivere alcune delle tecniche usate per interfacciare il microprocessore Z80.
- Parlare dei tre bus di un microcomputer basato sullo Z80.
- Parlare dei quattro importanti segnali di controllo che sono generati dalla CPU Z80 e che permettono alla stessa di interfacciarsi con l'esterno: $\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$, $\overline{IORQ}$.
- Parlare della configurazione dei pin di un microprocessore Z80.
- Essere in grado di leggere i diagrammi dei tempi che indicano come i fenomeni che avvengono all'interno dello Z80 devono essere sincronizzati con gli eventi che si verificano all'esterno.
- Dare la definizione di sincrono, dispositivo di I/O, CPU e memoria quando si riferiscono all'interfacciamento.

CHE COSA E' L'INTERFACCIAMENTO?

L'*interfacciamento* si può definire come l'associazione di membri di un gruppo (quali persone, strumenti, ecc.) in modo tale che essi siano in grado di funzionare in modo coordinato 17*.

* Vedi l'appendice E per tutti i riferimenti.

Ecco alcune definizioni di termini che verranno frequentemente usati nel corso del testo:

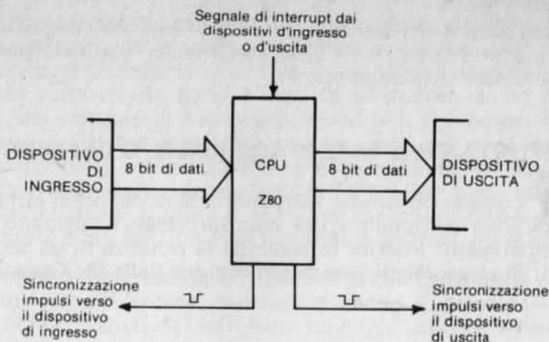
Sincrono	Quando viene applicato a due dispositivi o a due macchine significa al passo, in fase. Quando viene applicato ai calcolatori indica che l'esecuzione di una sequenza di operazioni è controllata per mezzo di segnali o di impulsi di clock (orologio).
Calcolatore Sincrono	Un calcolatore (computer) digitale in cui tutte le operazioni ordinarie vengono controllate da un clock principale.
Operazione Sincrona	Operazione di un sistema sotto il controllo di un clock.
Logica Sincrona	Il tipo di logica digitale usata in un sistema nel quale le operazioni vengono effettuate in sincronismo con gli impulsi di clock.
Sinc	Abbreviazione di sincrono, sincronizzazione, sincronizzare, ecc.
Sincronizzare	Mettere un elemento di un sistema al passo, in fase, con un altro.
Impulsi di Sincronizzazione	Impulsi originati dal dispositivo trasmittente ed inseriti nel dispositivo ricevente per fare sì che entrambi i dispositivi stiano al passo l'uno con l'altro.
Ingressi Sincroni	Quegli ingressi dei flip-flop che non controllano direttamente l'uscita, come si verifica per gli ingressi delle gate, ma solo quando il clock lo permette.

Le definizioni suddette sono state ricavate dalla bibliografia [2].. Quindi possiamo così definire che cosa vuol dire *interfacciare un calcolatore:*

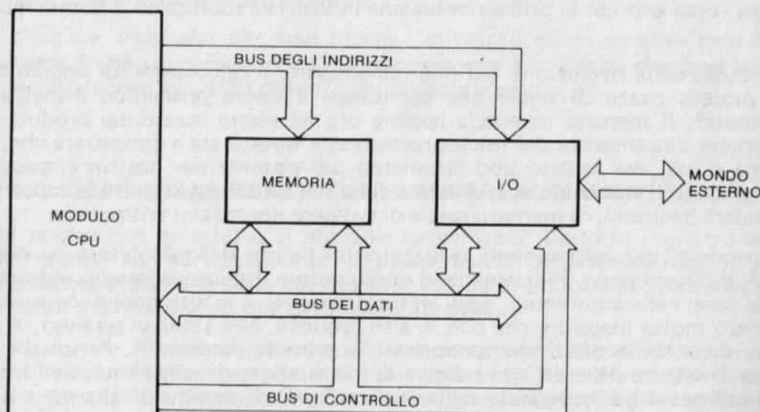
> Sincronizzare la trasmissione di dati digitali fra un calcolatore e dei dispositivi esterni, memoria e dispositivi di I/O compresi.

Sebbene i particolari dell'interfacciamento cambino a seconda del tipo di calcolatore impiegato, i principi generali dell'interfacciamento vengono applicati ad una vasta gamma di elaboratori. Per quanto riguarda il microprocessore Z80, gli obiettivi principali dell'interfacciamento si possono così riassumere (si veda anche la Figura 1–1):

- INGRESSO: Trasferire i dati da un dispositivo esterno al microprocessore.
- USCITA: Trasferire i dati dal microprocessore a un dispositivo esterno.
- GENERAZIONE DEGLI IMPULSI DI SINCRONIZZAZIONE: Generare gli impulsi che sincronizzano opportunamente il trasferimento dei dati in ingresso e in uscita, denominati *impulsi di selezione dispositivi,* in modo da coordinare le azioni del dispositivo esterno e del microprocessore.
- GESTIONE DELLE INTERRUZIONI: Rilevare ed osservare i segnali di interruzione (interrupt) che giungono al microprocessore dai dispositivi esterni.



(A) Diagramma di flusso.



(B) Diagramma di flusso "bus orientato".

Figura 1–1. I quattro compiti principali dell'interfacciamento: ingresso, uscita, generazione degli impulsi di selezione dispositivo, gestione delle interruzioni.

Il trasferimento dei dati fra la CPU ed un dispositivo esterno avviene sul bus dei dati. Il BUS DEI DATI dello Z80 è di otto bit ed è bidirezionale, il che significa che lo scambio delle informazioni avviene attraverso linee parallele che portano otto bit sia verso la CPU Z80 sia dalla stessa verso l'esterno. Il particolare dispositivo esterno che è coinvolto nel trasferimento dei dati che viene selezionato attraverso il BUS DEGLI INDIRIZZI. La CPU Z80 usa gli otto bit meno significativi del bus degli indirizzi (di 16 bit) per indirizzare in modo univoco un dispositivo esterno di I/O, mentre usa tutti i 16 bit dello stesso bus per indirizzare le locazioni di memoria. Il BUS DI CONTROLLO è formato da quei segnali che sincronizzano la presenza delle informazioni sul bus dei dati e su quello degli indirizzi con le attività della CPU e dei dispositivi esterni. Perciò nello scambio di informazioni fra la CPU ed i dispositivi esterni vengono coinvolti tre bus: il bus dei dati, il bus degli indirizzi ed il bus di controllo.
Lo scopo di questo capitolo e di quelli che seguono è quello di insegnarvi alcune

tecniche per interfacciare il Nanocomputer Z80 con altri dispositivi. Ma prima di entrare nei particolari riguardanti l'interfacciamento, vediamo perchè può interessarvi interfacciare il vostro microcomputer.

PERCHE' E' IMPORTANTE L'INTERFACCIAMENTO DEI MICROCOMPUTER?

Per rispondere a questa domanda, vorremmo citare dei brani estratti da un interessante articolo sull'interfacciamento dei microprocessori intitolato "La rivoluzione delle macchine intelligenti: inserire in prodotti la potenza di un cervello", comparso il 5 luglio 1976 su *Business Week*, una pubblicazione della Mc.Graw-Hill.

"Questa è la seconda rivoluzione industriale - dice Sidney Webb, vice-presidente esecutivo della TRW, Inc. Essa moltiplica il potere del cervello umano con la stessa forza con cui la prima rivoluzione industriale moltiplicò il potere muscolare.

Il motore della rivoluzione è il microprocessore, il calcolatore sul singolo chip, un piccolo pezzo di silicio che costituisce il cuore aritmetico e logico del computer. Il mercato comincia appena ora ad essere invaso dai prodotti che sfruttano l'intelligenza del microprocessore, e questo sta a dimostrare che, mai prima d'ora, era esistito uno strumento più potente per costruire macchine "intelligenti", macchine che, alle loro funzioni solite, aggiungono la capacità di prendere decisioni, di memorizzare e di svolgere operazioni aritmetiche.

"I prodotti più sorprendenti realizzati con l'aiuto del calcolatore su singolo chip, interesseranno direttamente il consumatore. I microprocessori entreranno nelle case, nelle automobili, negli elettrodomestici, e in altri beni di consumo in numero molto maggiore che non in altri prodotti. Nel 1980, vi saranno, in ogni casa, da sette a dieci microprocessori – prevede Andrew A. Perlowski, che dirige il settore delle attività relative ai microprocessori alla Honeywell Inc. La sua azienda è già impegnata nella realizzazione di sistemi di allarme e di gestione dell'energia per uso domestico.

"Nelle fabbriche, il calcolatore su singolo chip sta portando ad una tale riduzione del costo dell'intelligenza elettronica, che anche i prodotti più piccoli si trasformano in macchine intelligenti. Il microprocessore avvicina inoltre il giorno della realizzazione di fabbriche automatizzate, grazie alla possibilità di collegare le macchine intelligenti, i sensori ed altri strumenti ai sistemi distribuiti di controllo e di acquisizione dati".

"Il motivo dell'improvviso aumento delle vendite di microprocessori e dell'ondata di nuove macchine intelligenti a cui essi daranno origine, è semplicemente il prezzo. C. Lester Hogan, vice-presidente della Fairchild Camera and Instrument Corp., ne ha dato un'incisiva dimostrazione ad un convegno, tenuto a Boston alcune settimane fa. Egli si tolse di tasca 18 microprocessori e li mostrò al pubblico che lo ascoltava, dicendo: "Questo corrisponde ad una capacità di elaborazione che vale 18 milioni di dollari, o che per lo meno li valeva venti anni fa". Hogan ha poi spiegato che il suo microprocessore da 20 dollari aveva la stessa potenza del primo calcolatore commerciale, costruito dall'International Business Machines, e che all'inizio degli anni '50 costava un milione di dollari. "Quello che sto cercando di mettere in evidenza, ha detto Hogan, è che, oggi, la potenza dei calcolatori è praticamente gratuita".

Nello sviluppo dei prodotti basati su microprocessore:

"Il fatto di abbandonare le parti elettroniche convenzionali per usare le MPU porta ad una riduzione dei tempi di progettazione e dei costi di produzione, poichè, in tal modo, vengono sostituiti centinaia di circuiti integrati e di altre parti. Una volta che la MPU è inserita all'interno di un prodotto, essa può portare a dei vantaggi di marketing incredibili; si possono infatti alterare le funzioni di un prodotto non attraverso una nuova e costosa progettazione dei circuiti elettronici, ma semplicemente cambiando le istruzioni, cioè il software, memorizzato nella memoria della MPU. Con poca spesa, si possono aggiungere caratteristiche nuove, e le nuove macchine intelligenti sono così in grado di svolgere delle funzioni che risultavano prima troppo costose".

"Il software, oltre a rappresentare il problema più grosso che gli utenti della MPU si trovano ad affrontare, costituisce anche il principale problema di costo. Attualmente, i costi del software sono più alti per un microcomputer che per un minicomputer, dice Marley, un consulente del New Hampshire che ha realizzato 'prodotti intelligenti' per molte società. Egli dice di spendere fino a 100.000$ per ogni progetto software, mentre il costo dei progetti hardware si aggira intorno ai 20.000$.

Queste citazioni mostrano che ben presto i microprocessori saranno dappertutto. L'importante è che voi capiate che l'interfacciamento è la tecnica che deve essere applicata per trarre vantaggi dalla potenza dei microprocessori.

COME VIENE REALIZZATO L'INTERFACCIAMENTO?

In un precedente paragrafo, vi abbiamo brevemente illustrato i quattro concetti principali relativi all'interfacciamento: *ingresso, uscita, generazione degli impulsi di sincronizzazione e gestione delle interruzioni.* Abbiamo finora usato parecchi termini di cui daremo a questo punto una definizione formale.

Bus — Un percorso sul quale vengono trasferite le informazioni digitali, provenienti da una delle varie sorgenti e dirette ad una delle varie destinazioni. Si può verificare un solo trasferimento di informazioni alla volta. Mentre tale trasferimento ha luogo, tutte le altre sorgenti collegate al bus devono essere disabilitate.

Bus Bidirezionale dei Dati — Un bus di dati nel quale le informazioni digitali possono essere trasferite in entrambe le direzioni. Facendo riferimento al sistema microprocessore Z80, è il canale bidirezionale tramite il quale i dati vengono trasferiti fra la CPU, la memoria e gli altri dispositivi esterni.

Bus degli Indirizzi — Un bus unidirezionale sul quale appaiono le informazioni digitali che servono a identificare una particolare locazione di memoria o un particolare dispositivo di I/O. Il bus degli indirizzi del sistema microprocessore Z80 è costituito da 16 linee.

Indirizzo — Un gruppo di bit che identificano una specifica locazione di memoria o un dispositivo esterno. Un microprocessore Z80 usa 16 bit per indirizzare in modo univoco una locazione di memoria, mentre altri dispositivi esterni vengono identificati da un indirizzo a 8 bit.

Controllo — Quelle parti di un calcolatore che mettono a disposizione le istruzioni in sequenza opportuna, che interpretano le istruzioni stesse e che generano segnali di sincronizzazione opportuni.

Bus di Controllo Un insieme di linee su cui vengono presentati i segnali che regolano le operazioni di un microcomputer, della sua memoria e dei suoi dispositivi esterni. Questi segnali possono essere generati dalla CPU o da un dispositivo esterno. Il bus di controllo del sistema microcomputer Z80 è costituito da 13 linee e comprende i segnali che sincronizzano le operazioni di I/O tra la CPU e la memoria e gli altri dispositivi esterni, i segnali che controllano la CPU quali l'Interrupt, il segnale di attesa (WAIT) e di arresto (HALT), ed infine i segnali che controllano l'accesso al bus degli indirizzi e al bus di dati.

I/O Abbreviazione di Input/Output (ingresso/uscita).

Dispositivo di I/O Dispositivo di ingresso/uscita. Un lettore di schede, un'unità a nastro magnetico, una stampante o un dispositivo analogo che trasmette o riceve dati da un calcolatore o un dispositivo di memorizzazione secondario [2]. In senso più generale, qualunque dispositivo digitale, anche un solo circuito integrato, che trasmetta dati a un calcolatore, o che riceva dati o impulsi di strobe da un calcolatore.

CPU Abbreviazione di Central Processing Unit (Unità centrale di elaborazione, o di processo).

Unità Centrale di Elaborazione (grossi calcolatori) Detta anche elaboratore centrale. La parte di un calcolatore che contiene un'unità di controllo, l'unità aritmetica e logica, e gruppi di registri speciali. Essa controlla l'esecuzione delle istruzioni, effettua le operazioni aritmetiche e fornisce i segnali di temporizzazione, nonchè altre operazioni di gestione.

Unità Centrale di Elaborazione (microprocessori) Un solo circuito integrato che attua il trasferimento dei dati, il controllo, le operazioni aritmetiche, logiche e di gestione delle interruzioni, in conseguenza dell'esecuzione di istruzioni prelevate dalla memoria.

Memoria Qualunque dispositivo che possa memorizzare bit di livello logico 0 e 1, in modo che un singolo bit o un gruppo di bit (detto "parola") possano venire localizzati ed estratti dalla memoria stessa.

L'unità centrale di elaborazione Z80 (CPU)

Tenendo presenti le suddette definizioni, guardiamo ora l'architettura della Figura 1-1 nei particolari. Esaminiamo da vicino il modulo della CPU: per quanto riguarda il Nanocomputer, il modulo della CPU è formato da un solo circuito integrato a 40 pin, la CPU Z80. Nel Volume 1, abbiamo parlato del set di istruzioni che fa da supporto al microprocessore Z80. Nei paragrafi seguenti, sposteremo la nostra attenzione dal software all'hardware dello Z80. Vedremo prima uno schema funzionale del microprocessore Z80, e impareremo quindi la configurazione dei piedini del contenitore a doppia fila a 40 piedini che contiene il dispositivo elettronico Z80. Infine, ci interesseremo di alcuni diagrammi dei tempi che vi mostreranno esattamente come lo Z80 comunica con la memoria e con gli altri dispositivi esterni.

La Figura 1-2 mostra uno schema a blocchi funzionale della CPU Z80 che è riportato nel Manuale Tecnico della CPU Z80 pubblicato dalla SGS-ATES. Quando la CPU Z80 esegue un programma che risiede nella memoria ad essa associata, le istruzioni vengono lette una dopo l'altra dalla memoria stessa: l'indirizzo contenuto nel regi-

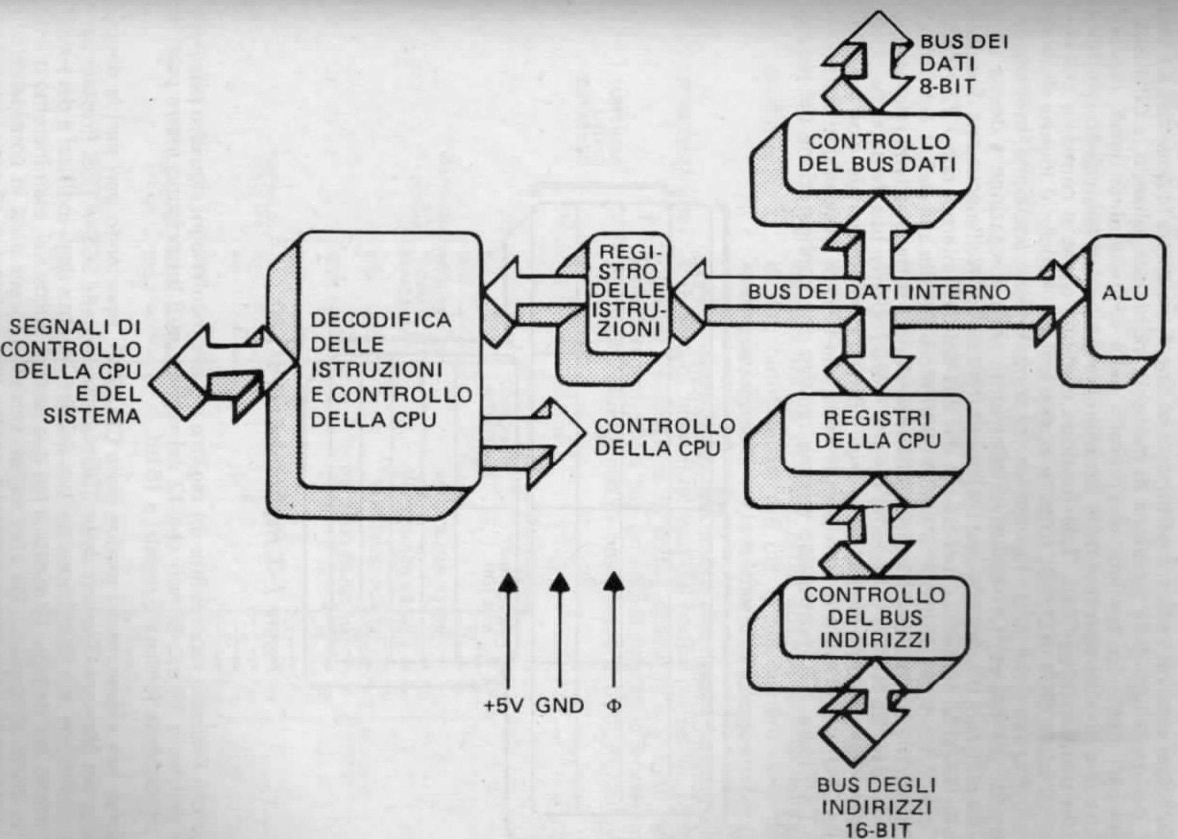


Figura 1-2. Schema a blocchi funzionale della CPU Z80.

stro PC viene posto sul bus degli indirizzi, sono generati e posti sul bus di controllo i segnali di controllo necessari per attivare la memoria, e sono infine trasferiti i dati presenti sul bus dei dati, nell'appropriato registro interno alla CPU. Ovviamente, le temporizzazioni sono un fattore determinante al fine di assicurare la presenza, sul bus dei dati, del contenuto della locazione di memoria indirizzata, quando la CPU legge il bus dei dati stessi. La funzione di controllo della CPU coordina questi compiti ed assicura che i codici operativi delle istruzioni vengano posti nel registro istruzioni e opportunamente decodificati. Tale funzione è adibita anche al controllo dell'esecuzione, da parte della ALU, di tutte le operazioni aritmetiche e logiche previste dal set di istruzioni dello Z80. Tali operazioni comprendono addizione, sottrazione, AND logico, OR logico, OR–esclusivo, confronto, shift e rotazione a destra ed a sinistra, incremento, decremento, set bit, reset bit e test bit. Nell'esecuzione di queste operazioni, la ALU comunica, per mezzo del bus dei dati interno, con i 22 registri interni, con il registro istruzioni, e con il controllore del bus dei dati. I controllori del bus dei dati e del bus degli indirizzi sovrintendono a tutte le attività relative allo scambio dei dati fra la CPU e l'esterno tramite i loro rispettivi bus. Notate che, mentre il bus dei dati è bidirezionale, il bus degli indirizzi è unidirezionale, cioè funziona solo dalla CPU verso l'esterno, non permettendo alla CPU di ricevere dati attraverso di esso. La Figura 1-3 mostra uno schema relativo alla configurazione dei registri della CPU.

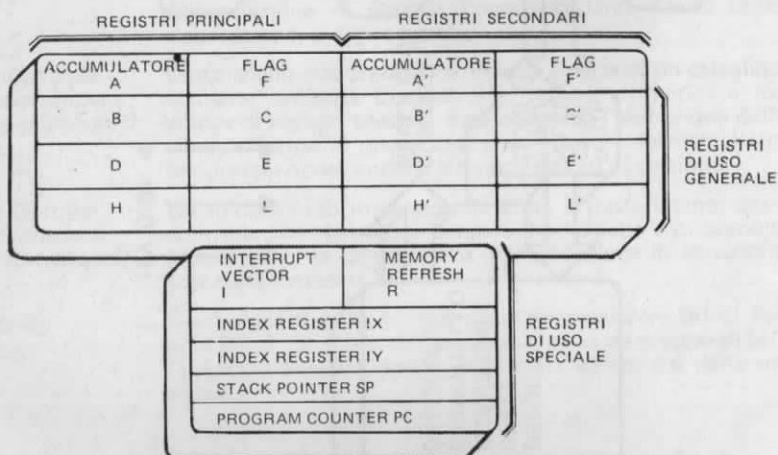


Figura 1-3. Registri della CPU Z80.

Ogni piccolo riquadro rappresenta un registro a 8 bit, mentre ogni riquadro più grande è un registro a 16 bit. Si noti che 12 dei registri ad 8 bit possono essere usati in coppie, in modo da formare 6 registri a 16 bit.

Passiamo ora a descrivere i piedini della CPU Z80, riportando pari pari la descrizione data nel Manuale Tecnico dello Z80 pubblicato dalla SGS-ATES. Prestate particolare attenzione alla descrizione del bus dei dati, del bus degli indirizzi e dei segnali di controllo del sistema. In questo e nei due capitoli successivi, esamineremo questi segnali in modo dettagliato. Gli altri segnali verranno invece presi in considerazione più avanti in modo più o meno particolareggiato. Non aspettatevi di capire completamente tale descrizione. Il fatto di capire queste informazioni, e una volta arrivati al termine del libro, metterle in pratica in applicazioni pratiche riguardanti l'interfacciamento dello Z80, è proprio uno degli scopi di questo volume. Il nostro obiettivo nel

fornirvi ora queste informazioni è quello di darvi un'idea di quanto seguirà dopo e di imprimervi chiaro in mente che lo Z80 è semplicemente un circuito integrato a 40 piedini (si veda la Figura 1-4) e non qualcosa di cui avere paura.

Vi descriviamo ora la funzione di ogni pin:

A0-A15 (Bus degli indirizzi) — Uscita tri-state, attivi alti. A0-A15 formano un bus degli indirizzi a 16 bit. Tale bus fornisce l'indirizzo per lo scambio dei dati con la memoria (fino a 64K byte = 2^{16}) e per lo scambio dei dati con i dispositivi di I/O. Per indirizzare l'I/O si utilizzano gli 8 bit di indirizzo più bassi e l'utente può così selezionare direttamente fino a 256 = 2^8 porte di ingresso o altrettante porte di uscita. A0 è il bit di indirizzo meno significativo. Mentre si verifica il refresh, i 7 bit più bassi contengono un indirizzo di refresh valido (argomento discusso nel Capitolo 3).

D0-D7 (Bus dei dati) — Ingresso/uscita tri-state, attivi alti, D0-D7 formano un bus di dati bidirezionale a 8 bit. Il bus dei dati viene usato per lo scambio dei dati con la memoria ed i dispositivi di I/O.

$\overline{\text{M1}}$ (Ciclo macchina uno) — Uscita, attivo basso. $\overline{\text{M1}}$ indica che il ciclo macchina in corso è il ciclo di fetch (prelievo) del codice operativo. Notate che, durante l'esecuzione di codici operativi a 2 byte, $\overline{\text{M1}}$ viene generato ogni volta che un codice operativo è prelevato dalla memoria. Questi codici operativi a 2 byte iniziano sempre

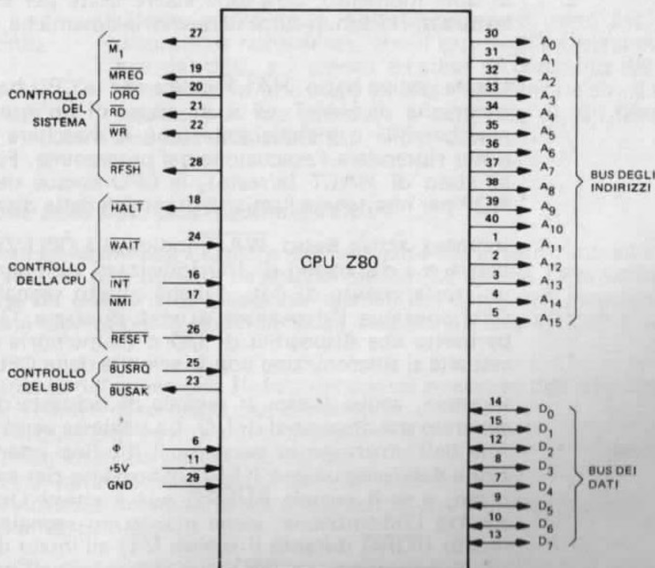


Figura 1-4. Configurazione dei pin della CPU Z80.

con CB, DD, ED, FD (esadecimali). Inoltre la presenza contemporanea di $\overline{M1}$ e di $\overline{IORQ}$ sta ad indicare un ciclo di riconoscimento dell'interruzione.

$\overline{MREQ}$
(Richiesta di memoria)

Uscita tri-state, attivo basso. Il segnale di richiesta di memoria indica che il bus degli indirizzi contiene un indirizzo valido per un'operazione di lettura o di scrittura in memoria.

$\overline{IORQ}$
(Richiesta di ingresso/uscita)

Uscita tri-state, attivo basso. Il segnale di $\overline{IORQ}$ indica che la metà inferiore del bus degli indirizzi contiene un indirizzo di I/O valido per un'operazione di lettura o di scrittura in I/O. Inoltre, il segnale di $\overline{IORQ}$ viene generato insieme al segnale $\overline{M1}$ nel riconoscimento di un'interruzione, per indicare che si può porre sul bus dei dati un vettore di risposta all'interruzione. Le operazioni di riconoscimento dell'interruzione avvengono quando il segnale $\overline{M1}$ è attivo, condizione che non si verifica per le operazioni di I/O.

$\overline{RD}$
(Lettura in memoria)

Uscita tri-state, attivo basso. $\overline{RD}$ sta ad indicare che la CPU vuole leggere i dati contenuti in memoria o in un dispositivo di I/O. La memoria o il dispositivo di I/O indirizzati devono usare questo segnale per porre i dati sul bus dei dati della CPU.

$\overline{WR}$
(Scrittura in memoria)

Uscita tri-state, attivo basso. $\overline{WR}$ sta ad indicare che il bus dei dati della CPU contiene dati validi da memorizzare nella memoria o nel dispositivo di I/O indirizzati.

$\overline{RFSH}$
(Refresh)

Uscita, attivo basso. $\overline{RFSH}$ indica che i 7 bit inferiori del bus degli indirizzi contengono un indirizzo di refresh (rinfresco) per le memorie dinamiche e che il segnale $\overline{MREQ}$, attivo in quel momento, dovrebbe essere usato per effettuare una lettura di refresh di tutte le memorie dinamiche.

$\overline{HALT}$
(Stato di halt)

Uscita, attivo basso. $\overline{HALT}$ indica che la CPU ha eseguito una istruzione di HALT ed è in attesa di un'interruzione non mascherabile o mascherabile (con la maschera abilitata) per poter riprendere l'esecuzione del programma. Finchè perdura lo stato di HALT (arresto), la CPU esegue delle istruzioni NOP per mantenere l'attività di refresh della memoria.

$\overline{WAIT}$
(Attesa)

Ingresso, attivo basso. $\overline{WAIT}$ indica alla CPU Z80 che la memoria o i dispositivi di I/O indirizzati non sono pronti per un trasferimento di dati. Finchè questo segnale è attivo, la CPU continua l'attuazione di stati di attesa. Questo segnale permette che dispositivi di I/O o di memoria di qualunque velocità si sincronizzino con la velocità della CPU.

$\overline{INT}$
(Richiesta d'interruzione)

Ingresso, attivo basso. Il segnale di richiesta d'interruzione generato dai dispositivi di I/O. La richiesta verrà accettata alla fine dell'istruzione in corso se il flip-flop interno di abilitazione dell'interruzione (IFF), controllato dal software è abilitato, e se il segnale $\overline{BUSRQ}$ non è attivo. Quando la CPU accetta l'interruzione, viene inviato un segnale di riconoscimento ($\overline{IORQ}$ durante il tempo $\overline{M1}$) all'inizio del ciclo istruzione successivo. La CPU può rispondere all'interruzione in tre modi diversi, dei quali parleremo dettagliatamente più avanti.

$\overline{NMI}$
(Interruzione non mascherabile)

Ingresso, attivo sul fronte negativo. La linea di richiesta d'interruzione non mascherabile ha una priorità superiore rispetto a $\overline{INT}$ e viene sempre riconosciuta alla fine dell'istruzione in corso, a prescindere dallo stato del flip-flop di abilitazione dell'interruzione. $\overline{NMI}$ costringe automaticamente la CPU Z80 a riprendere l'esecuzione della locazione 0066 esadecimale. Il contatore di programma viene automaticamente salvato nello stack esterno, in modo che l'utente possa successivamente riprendere il programma interrotto. Notate che l'esecuzione continuata di cicli WAIT può impedire all'istruzione in corso di terminare, e che un $\overline{BUSRQ}$ sarà prioritario rispetto ad un $\overline{NMI}$.

$\overline{RESET}$
(Azzeramento)

Ingresso, attivo basso. $\overline{RESET}$ forza il contatore di programma a zero ed inizializza la CPU. Inizializzare la CPU significa:
a) Disabilitare il flip-flop di abilitazione dell'interruzione
b) Caricare il registro I con 00 esadec.
c) Caricare il registro R con 00 esadec.
d) Predisporre il modo interrupt 0.
Durante il reset, il bus degli indirizzi e il bus dei dati si pongono in uno stato di alta impedenza e tutti i segnali di uscita di controllo diventano inattivi.

$\overline{BUSRQ}$
(Richiesta del bus)

Ingresso, attivo basso. Il segnale di richiesta del bus viene usato per richiedere al bus degli indirizzi, al bus dei dati, e alle uscite tri-state di controllo della CPU, di porsi in uno stato di alta impedenza, in modo che questi bus possono venire controllati da altri dispositivi. Quando $\overline{BUSRQ}$ viene attivato, la CPU pone i bus in uno stato di alta impedenza, non appena termina il ciclo macchina in corso.

$\overline{BUSAK}$
(Riconoscimento del bus)

Uscita, attivo basso. Questo segnale è usato per indicare al dispositivo richiedente, che il bus degli indirizzi della CPU, il bus dei dati, e i segnali tri-state di controllo del bus, sono stati posti in uno stato di alta impedenza e che il dispositivo esterno può ora prendere il controllo di tali linee. Clock a una fase di livello TTL.

I cicli istruzione dello Z80: cicli macchina e cicli T

Passiamo ora ad esaminare l'insieme dei fenomeni che si verificano all'interno dello Z80 e sui suoi 40 pin, durante l'esecuzione di un'istruzione. Ogni istruzione dello Z80 è formata da una serie di operazioni elementari, dette *cicli macchina*. Vi sono solamente sette operazioni elementari (cicli macchina) che lo Z80 è in grado di eseguire:

1. Prelevamento dalla memoria (fetch) del codice operativo dell'istruzione
2. Ciclo di lettura o di scrittura dati in memoria
3. Ciclo di lettura o di scrittura in dispositivi di I/O
4. Ciclo di richiesta/riconoscimento del bus
5. Ciclo di richiesta/riconoscimento dell'interruzione
6. Ciclo di richiesta/riconoscimento dell'interruzione non mascherabile
7. Uscita dall'istruzione Halt.

Chiaramente le prime sei operazioni hanno un riferimento diretto con i quattro obiettivi principali dell'interfacciamento. Nella parte restante di questo capitolo, e nei capitoli successivi, ci limiteremo a trattare i primi tre tipi di cicli macchina. Eseguen-

do un'istruzione del tipo LD A, 00H (esadecimale: 3E 00) memorizzata nella locazione di memoria 0100, lo Z80 esegue per prima cosa un ciclo M1, in modo da prelevare dalla memoria il codice operativo dell'istruzione. Questo avviene ponendo l'indirizzo, 0100, sul bus degli indirizzi a 16 bit ed attivando i segnali di controllo $\overline{MREQ}$ e $\overline{RD}$ (pin 19 e 21). La memoria interpreta questi segnali come segue:

a) $\overline{MREQ}$ attivo basso (0 logico) significa che viene attuato, in quel momento, un accesso in memoria.

b) $\overline{RD}$ attivo basso (0 logico) significa che l'accesso è una operazione di lettura. 0100 sul bus degli indirizzi indica qual'è la locazione di memoria che deve essera letta.

La memoria risponde ponendo il contenuto della locazione 0100 sul bus dei dati. Quindi la CPU legge il bus dei dati e ne memorizza il contenuto. Quando l'operazione di lettura riguarda il primo byte di un'istruzione, viene attivato il segnale $\overline{M1}$, in modo da indicare che il byte deve essere un codice operativo. Per le istruzioni con due codici operativi, per ottenere i codici della memoria, vengono eseguiti due cicli M1. Nel nostro esempio si verifica un solo ciclo M1, per leggere il codice operativo 3E.

Lo Z80 decodifica il codice 3E come una istruzione di caricamento immediato nell'accumulatore ed esegue quindi un ciclo macchina di lettura in memoria per caricare il contenuto della locazione 0100 nell'accumulatore. Una lettura in memoria è quasi identica a un ciclo M1, a parte il fatto che il segnale $\overline{M1}$ non viene attivato e le temporizzazioni sono leggermente diverse.
Prima di mostrarvi i diagrammi dei tempi per il ciclo macchina M1, dobbiamo parlarvi dei cicli T. Ogni ciclo macchina è a sua volta suddiviso in alcuni cicli T. I cicli T hanno una corrispondenza uno a uno con gli impulsi provenienti dall'ingresso di clock Φ (pin 6) dello Z80. La massima velocità di clock alla quale può funzionare uno Z80 selezionato è di 4 megahertz, cioè 4.000.000 cicli al secondo. Ciascun ciclo M1 dura perciò 250 nanosecondi. Il numero di cicli T per ogni ciclo macchina è legato alla funzione del ciclo macchina stesso. Piú la funzione è complicata, più aumenta il numero di cicli T. Un ciclo macchina M1 è formato da quattro cicli T, mentre per una lettura in memoria ne sono sufficienti tre. Il ciclo M1 è più complicato poichè durante questo ciclo devono essere decodificate le istruzioni.

Quindi, riassumendo, le istruzioni sono costituite da cicli macchina, che a loro volta sono formati da cicli T. Il primo byte di ogni istruzione è un byte di codice operativo, perciò lo Z80 esegue sempre almeno un ciclo M1 per ogni istruzione. Mentre esistono diversi cicli macchina, c'è un solo tipo di ciclo T, che è in effetti un ciclo della durata di 250 nanosecondi (a 4 MHz) corrispondente al passaggio del clock da 0 a 1 e poi ancora a 0 (si veda la Figura 1-5).

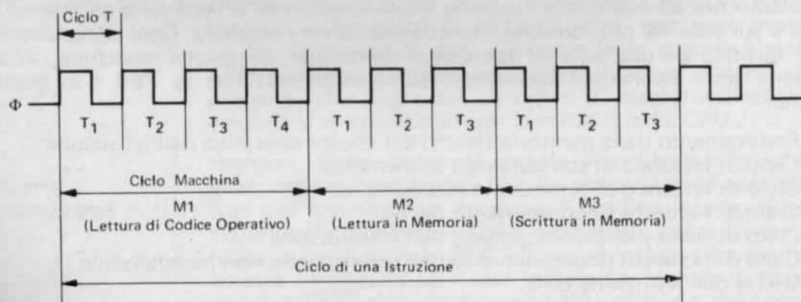


Figura 1-5. Suddivisione di una istruzione dello Z80 in cicli macchina e cicli T.

Diagramma dei tempi CPU Z80

La Figura 1-6 mostra il diagramma dei tempi della CPU Z80 per un ciclo M1 e cioè, per un ciclo di prelievo di un codice operativo. Senza che sia data indicazione quantitativa dei tempi, vi vengono mostrate le relazioni temporali che esistono tra i segnali di Indirizzo, Dati e Controllo, attivi durante un ciclo M1.

Per comprendere in modo esatto le relazioni temporali, che possono essere determinanti all'interno di un sistema a microcomputer, studieremo in dettaglio la successiva Figura 1-7. I tempi indicati in questa Figura *non* sono relativi ai segnali presenti ai pin della CPU Z80, ma quelli dei segnali cui potete collegarvi attraverso i vari bus, in ingresso e in uscita dal Nanocomputer. I nomi dei segnali sono gli stessi, con la sola differenza di essere preceduti da un prefisso "B". Questa "B" indica che i segnali non sono quelli della CPU Z80 (per es. le linee di indirizzo A0-A15) ma segnali bufferizzati (per es. le linee di indirizzo bufferizzate BA0-BA15). La doppia nomenclatura è necessaria in quanto bufferizzazione ed altre operazioni logiche effettuate sulla scheda, aggiungono dei ritardi (dell'ordine di 30-100 nanosecondi) ai vari segnali, ed in certi casi questo provoca differenze apprezzabili.

E' per voi molto importante imparare a leggere i diagrammi dei tempi come questo, in quanto sono il mezzo più significativo utilizzato dai costruttori e dai progettisti digitali per mostrare come i circuiti e/o i singoli chip operano. In questa sezione esamineremo in dettaglio il diagramma dei tempi relativo al ciclo M1, e vi mostreremo come leggerlo. Tutti i concetti che apprenderete saranno applicabili anche alla lettura delle temporizzazioni degli altri cicli macchina dello Z80 e del nanocomputer presentati nei capitoli seguenti.

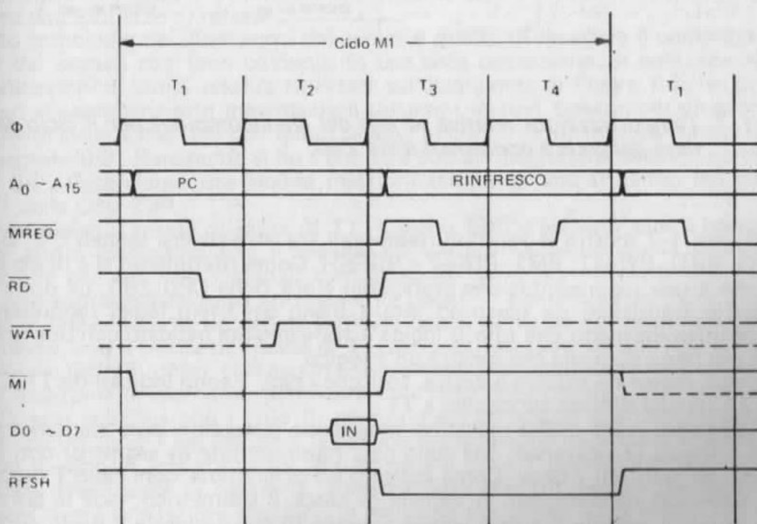


Figura 1-6. Temporizzazione della CPU Z80 nel ciclo M1 (prelievo del codice operativo).

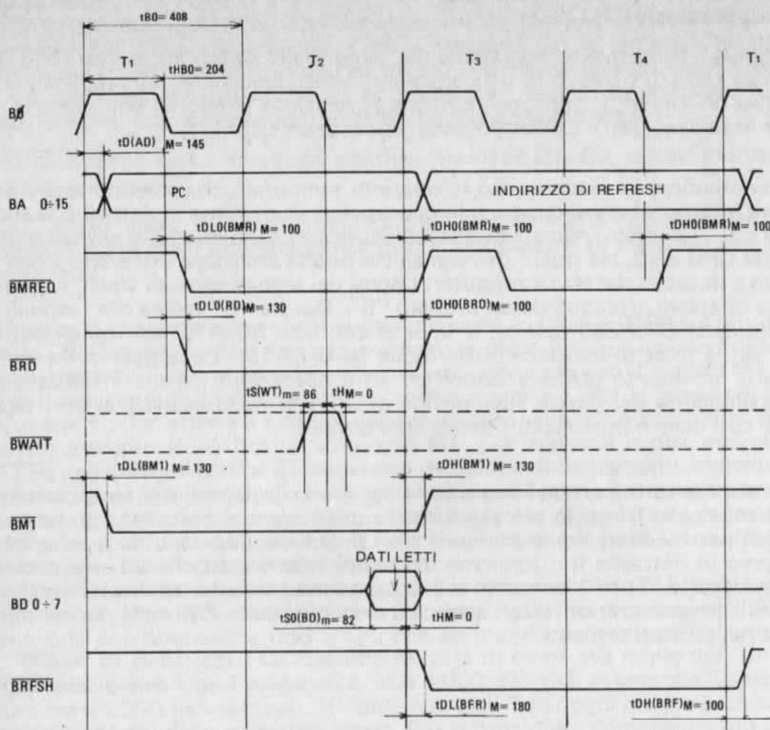


Figura 1-7. Temporizzazioni relative al bus del Nanocomputer per il ciclo M1 (prelievo del codice operativo) dello Z80.

La Figura 1-7 mostra le relazioni temporali tra otto diversi segnali BΦ, BA0-15, $\overline{\text{BMREQ}}$, $\overline{\text{BRD}}$, $\overline{\text{BWAIT}}$, $\overline{\text{BM1}}$, $\overline{\text{BD0-7}}$ e $\overline{\text{BRFSH}}$. Come riferimento si è preso il segnale BΦ, che è una rappresentazione grafica del clock della CPU Z80. Le onde quadre indicano le transizioni da basso-ad alto-a basso dei livelli logici rappresentate in logica positiva, in modo che uno 0 logico è graficamente indicato con una linea orizzontale più bassa di quella che mostra un 1 logico.

Il tempo cresce da sinistra a destra, cosicchè i cicli T sono indicati da T1, T2, etc., dove T2 è temporalmente successivo a T1.

Il diagramma vuole essere realistico, mostrando transizioni non istantanee da 0 logico ad 1 logico (e viceversa), che sono così rappresentate da segmenti non verticali colleganti gli stati alti e bassi. Come indicato sul diagramma, ogni ciclo T dura esattamente tBΦ=408 nanosecondi. Il segnale di clock è simmetrico, cioè la porzione di tempo in cui il segnale è alto è uguale a quella in cui il segnale è basso ed è pari a 204 nanosecondi (ns). L'intero diagramma mostra quattro cicli T e quindi rappresenta una finestra di 1632 ns di tempo operativo della CPU Z80.

La rappresentazione grafica del bus degli indirizzi, BA0...BA15, può a prima vista, dare adito a confusione, perchè non è costituita da una sola linea, come nel caso di BΦ.

Questo è dovuto al fatto che il grafico deve rappresentare 16 linee, da BA0 a BA15, anzichè un solo piedino della CPU Z80.

Teoricamente dovrebbe esserci una corrispondenza uno ad uno tra i segnali e le loro rappresentazioni grafiche in un diagramma dei tempi; fatta eccezione per i pin degli indirizzi e dei dati, questo è quasi sempre vero. Tuttavia, per gli indirizzi ed i dati, è stata adottata la rappresentazione che potete vedere in Figura 1-7, per i seguenti motivi:

a) E' estremamente scomodo rappresentare ogni segnale di indirizzo o di dato con una linea separata;

b) E' più conveniente e spesso più significativo, disegnare una doppia linea, come potete vedere nella Figura 1-7, e porre una didascalia che indichi il dato o l'indirizzo presente su tali linee al tempo mostrato;

c) Le "X", che si formano quando le due linee si incrociano, delimitano il tempo per cui il valore indicato è presente sulle linee dei dati o degli indirizzi.

Il diagramma dei tempi della Figura 1-7 mostra che i 16 pin di indirizzo presentano dapprima il contenuto del registro PC, e poi un indirizzo di refresh, durante un ciclo M1. Una notazione addizionale indica che $tD\ (AD)_M = 145$. Questo significa che il tempo intercorrente tra il fronte di salita di T1(quando $B\Phi$ diventa 1)e il momento in cui PC è presente sul bus degli indirizzi non supera i 145 ns.

Nel diagramma le linee verticali sottili, che indicano i punti di transizione nelle oscillazioni di $B\Phi$, sono tracciate per aiutarvi ad individuare i punti significativi del grafico. Quando il contenuto del PC viene posto sulle linee di indirizzo, è ben più interessante conoscere, invece che le temporizzazioni mostrate, l'istante in cui questo valore è effettivamente stabile sulle linee di indirizzo. L'informazione diventa stabile (cioè valida) sempre alcuni nanosecondi dopo essere stata posta sulle linee.

Addirittura in Figura 1-7, non è dato l'istante esatto in cui il contenuto del PC è sostituito dall'indirizzo di refresh.

Scopo principale dei diagrammi dei tempi è quello di mostrare il comportamento *relativo* dei segnali che sono coinvolti in una data operazione. Si noti, che senza le varie indicazioni di tempi assoluti riportate sul diagramma di Figura 1-7, nel diagramma stesso vi sarebbero solo informazioni di tempi relativi. Spesso, per un progettista è sufficiente avere a che fare con tempi relativi in cui il riferimento sia sempre il clock, cioè il segnale $B\Phi$. Raramente si ha a che fare con un diagramma che non contenga il segnale $B\Phi$. Osserviamo ora alcune relazioni temporali che si hanno nel caso del ciclo M1 della CPU Z80.

Appena dopo il fronte di salita di T1, il segno $\overline{BM1}$ è attivato, cioè è basso (stato normale alto). (Sappiamo che il segnale $\overline{BM1}$ è normalmente alto a seguito della convenzione per cui tutti i segnali contrassegnati sono normalmente alti, mentre i segnali non soprassegnati sono normalmente bassi). Il contenuto del PC è poi posto sulle sedici linee BA0-15 (in genere indicate con il nome di *Bus degli Indirizzi*).Successivamente, con il fronte di discesa di T1, sono attivati i segnali $\overline{BMREQ}$ e $\overline{BRD}$.

Un intero ciclo T, dopo, ci si aspetta che la memoria risponda ponendo il contenuto della locazione di memoria specificata sulle otto linee dei dati BD0-7 (il *Bus dei Dati)*. Questo significa che i chip di memoria del computer e la relativa circuiteria hanno circa un ciclo T per decidere la particolare locazione indirizzata e per accedere agli 8 bit di dato in essa contenuti. Più corto è il ciclo T (più veloce cioè è il clock), più veloce deve essere la memoria. Naturalmente occorre sempre pagare qualcosa in più per avere prestazioni migliori: le memorie più veloci sono infatti più care di quelle lente. Il segnale $\overline{BM1}$ è disattivato (posto alto) dopo il fronte di salita di T3.

In questo istante, i dati della memoria sono già stati letti dalla CPU, e quindi possono essere tolti dal bus dei dati. Fatto tutto ciò, inizia (ma è ancora parte dello stesso ciclo M1) una nuova operazione, detta *operazione di refresh*. Osservate che il segnale $\overline{BRFSH}$ non è attivato finchè non si è verificato il fronte di salita di T3; con-

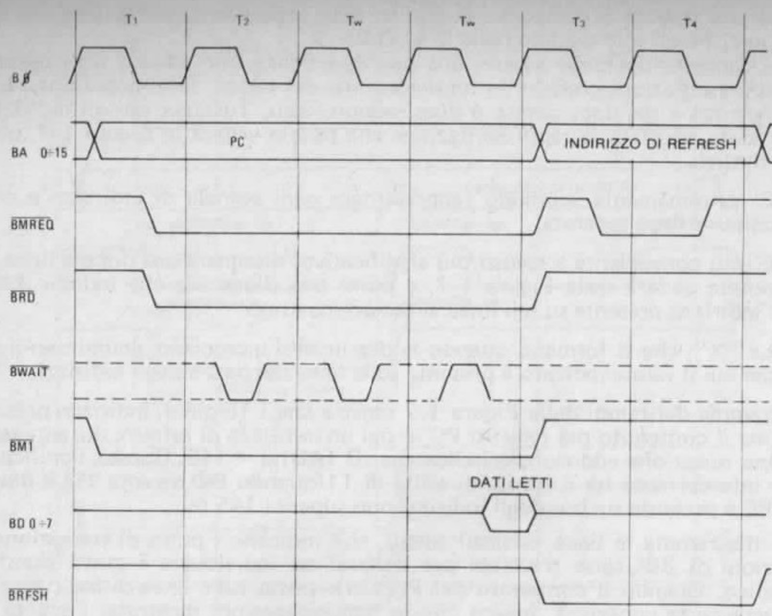


Figura 1–8. Temporizzazioni relative al bus del Nanocomputer per il ciclo M1, con inseriti due tasti T_W.

temporaneamente viene posto un indirizzo di refresh (16 bit) sul bus degli indirizzi.

Non discuteremo qui l'operazione di refresh; evidenziamo unicamente che tale operazione ha luogo solo durante i cicli M1. Dovreste accorgervi che vi è ancora un segnale di cui non abbiamo parlato, il $\overline{\text{BWAIT}}$. Il pin $\overline{\text{BWAIT}}$ sul bus del Nanocomputer, è direttamente collegato ai pin 24 ($\overline{\text{WAIT}}$) della CPU Z80 ed è un pin di ingresso, campionato dalla CPU solo in certi particolari istanti. Al di fuori di tali istanti lo stato di questo pin $\overline{\text{WAIT}}$ non è significativo.

Le linee tratteggiate indicano sul grafico i periodi di tempo "non significativi". La CPU va a leggere, durante un ciclo M1, il pin $\overline{\text{WAIT}}$ solo al fronte di discesa del clock durante T2, se il pin è alto, il segnale è inattivo, e la CPU procede nella normale sequenza operativa del ciclo M1. Se viceversa il segnale $\overline{\text{BWAIT}}$ è basso (attivo), vuol dire che occorre inserire uno stato di WAIT nel ciclo M1. Gli stati di WAIT sono usati per estendere il tempo, a disposizione della memoria per rispondere ad una richiesta di accesso a più di un ciclo T.

Per ogni stato di WAIT (attesa) inserito nell'esecuzione di un ciclo M1, la memoria può contare su un ciclo T addizionale per rispondere.

Durante il fronte di discesa del clock di ciascun ciclo T addizionale, la CPU verifica ancora il pin $\overline{\text{WAIT}}$ per vedere se è stato disattivato. Se non è così, viene aggiunto un altro stato di wait. Se invece il segnale di WAIT è ritornato alto, la CPU procede a leggere il dato dal bus dei dati ed effettua un'operazione di refresh. La Figura 1-8

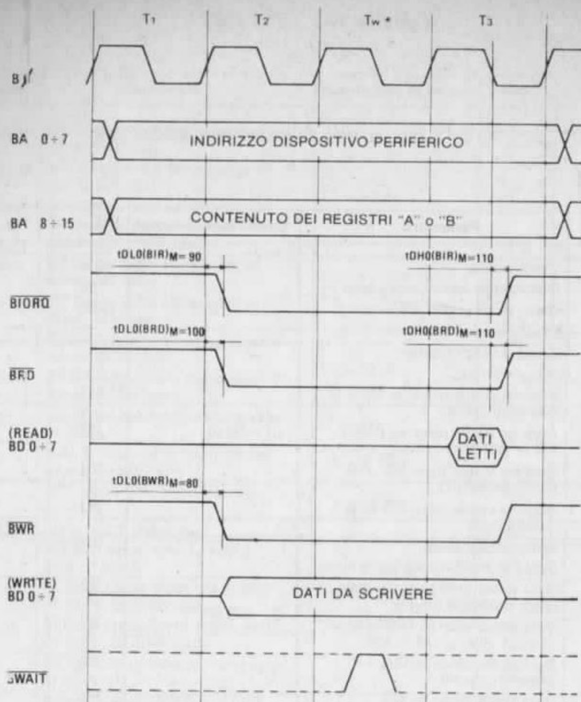


Figura 1-9. Temporizzazioni relative al bus del Nanocomputer per il ciclo di uscita dello Z80 verso un dispositivo esterno.

mostra un ciclo M1 in cui sono stati inseriti due stati di wait. Si noti come il segnale $\overline{BWAIT}$ è basso in corrispondenza di due fronti di discesa di $B\Phi$; successivamente torna alto. Si noti che per il resto del tempo lo stato di $\overline{BWAIT}$ è non significativo, perchè la CPU non lo campiona.

Vediamo ora i diagrammi di tempo di un ciclo di uscita, non verso la memoria, ma verso un dispositivo esterno.

Supponiamo di incontrare l'istruzione OUT (01H), A (esadecimale D3 01) nel corso dell'esecuzione di un programma dello Z80. Questa istruzione fa sì che lo Z80 ponga, in uscita, sulla "porta 01" il contenuto dell'accumulatore. Nei prossimi capitoli parleremo molto più dettagliatamente di questa istruzione. Dopo avere eseguito il ciclo M1, per leggere il codice operativo D3 dalla memoria e decodificarne il significato, la CPU sa di avere un'istruzione di uscita a 2 byte, nella quale il secondo byte sarà il numero (l'indirizzo) della porta. Viene così iniziano un ulteriore ciclo di lettura in memoria per leggere la locazione di memoria successiva e inserire così lo 01 nella CPU. Il ciclo successivo eseguito dalla CPU è il ciclo di uscita illustrato nella Figura 1-9. Notate che i segnali di controllo utilizzati sono $\overline{BWR}$ e $\overline{BIORQ}$. L'indirizzo della porta (01) viene posto sugli otto bit più bassi del bus degli indirizzi, BA0-7, mentre il contenuto dell'accumulatore viene posto sul bus dei dati, BD0-7. Sia il bus dei dati che il bus degli indirizzi contengono l'informazione corretta quando vengono attivati i due segnali di controllo $\overline{BWR}$ e $\overline{IORQ}$. E' questa una situazione critica,

Tabella 1-1. CPU Z80

(T_{amb}= da 0°C a 70°C, V_{CC}= +5V ± 5%, se non diversamente specificato)

Z80 CPU

Segnale	Simbolo	Parametro	Condizioni di misura	Min.	Tip.	Max.	Unità
Φ	t_c	Clock period		0.4		[12]	μsec
	$t_{w(\Phi H)}$	Clock pulse width, clock high		180		[E]	nsec
	$t_{w(\Phi L)}$	Clock pulse width, clock low		180		2000	nsec
	$t_{r,f}$	Clock rise and fall time				30	nsec
A_{0-15}	$t_{D(AD)}$	Address output delay	C_L = 50 pF			145	nsec
	$t_{F(AD)}$	Delay to float				110	nsec
	t_{acm}	Address stable prior to $\overline{MREQ}$ (Memory cycle)		[1]			nsec
	t_{aci}	Address stable prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O cycle)		[2]			nsec
	t_{ca}	Address stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$		[3]			nsec
	t_{caf}	Address stable from $\overline{RD}$ or $\overline{WR}$ During float		[4]			nsec
D_{0-7}	$t_{D(D)}$	Data output delay	C_L = 50 pF			230	nsec
	$t_{F(D)}$	Delay to float during write cycle				90	nsec
	$t_{S\Phi(D)}$	Data setup time to rising edge of clock during M1 cycle		50			nsec
	$t_{S\overline{\Phi}(D)}$	Data setup time to falling edge of clock during M2 to M5		60			nsec
	t_{dcm}	Data stable prior to $\overline{WR}$ (memory cycle)		[5]			nsec
	t_{dci}	Data stable prior to $\overline{WR}$ (I/O cycle)		[6]			nsec
	t_{cdf}	Data stable from $\overline{WR}$		[7]			
	t_H	Any hold time for setup time		0			nsec
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay from falling edge of clock, $\overline{MREQ}$ low	C_L = 50 pF			100	nsec
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay from rising edge of clock, $\overline{MREQ}$ high				100	nsec
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay from falling edge of clock, $\overline{MREQ}$ high				100	nsec
	$t_{w(\overline{MRL})}$	Pulse width, $\overline{MREQ}$ low		[8]			nsec
	$t_{w(\overline{MRH})}$	Pulse width, $\overline{MREQ}$ high		[9]			nsec
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay from rising edge of clock, $\overline{IORQ}$ low	C_L = 50 pF			90	nsec
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from falling edge of clock, $\overline{IORQ}$ low				110	nsec
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay from rising edge of clock, $\overline{IORQ}$ high				100	nsec
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from falling edge of clock, $\overline{IORQ}$ high				110	nsec
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay from rising edge of clock, $\overline{RD}$ low	C_L = 50 pF			100	nsec
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from falling edge of clock, $\overline{RD}$ low				130	nsec
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay from rising edge of clock, $\overline{RD}$ high				100	nsec
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from falling edge of clock, $\overline{RD}$ high				110	nsec

Segnale	Simbolo	Parametro	Condizioni di misura	Min.	Tip.	Max.	Unità
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay from rising edge of clock, $\overline{WR}$ low	$C_L = 50$ pF			80	nsec
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay from falling edge of clock, $\overline{WR}$ low				90	nsec
	$t_{DH\overline{\Phi}(WR)}$	WR Delay from falling edge of clock, $\overline{WR}$ high				100	nsec
	$t_{w(\overline{WRL})}$	Pulse width, $\overline{WR}$ low		[10]			nsec
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay from rising edge of clock, $\overline{M1}$ low	$C_L = 50$ pF			130	nsec
	$t_{DH(M1)}$	$\overline{M1}$ Delay from rising edge of clock, $\overline{M1}$ high				130	nsec
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay from rising edge of clock, $\overline{RFSH}$ low	$C_L = 50$ pF			180	nsec
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay from rising edge of clock, $\overline{RFSH}$ high				150	nsec
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ setup time to falling edge of clock		70			nsec
$\overline{HALT}$	$t_{D(HT)}$	HALT Delay time from falling edge of clock	$C_L = 50$ pF			300	nsec
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ setup time to rising edge of clock		80			nsec
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse width, $\overline{NMI}$ low		80			nsec
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ setup time to rising edge of clock		80			nsec
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay from rising edge of clock, $\overline{BUSAK}$ low	$C_L = 50$ pF			120	nsec
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay from falling edge of clock, $\overline{BUSAK}$ high				110	nsec
RESET	$t_{s(RS)}$	$\overline{RESET}$ setup time to rising edge of clock		90			nsec
	$t_{F(C)}$	Delay to float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)				100	nsec
	t_{mr}	$\overline{M1}$ stable prior to $\overline{IORQ}$ (Interrupt Ack.)		[11]			nsec

NOTE:

A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.
B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.
D. Output delay vs. loaded capacitance: $T_{amb} = 70°C$ $V_{CC} = +5V \pm 5\%$.
Add 10 nsec delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus & 100 pF for address & control lines.
E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum.

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$
[2] $t_{aci} = t_c - 80$
[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$
[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$
[5] $t_{dcm} = t_c - 210$
[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$
[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$
[8] $t_{w(MRL)} = t_c - 40$
[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$
[10] $t_{w(\overline{WR}L)} = t_c - 40$
[11] $t_{mr} = 2 t_c + t_{w(\Phi H)} + t_f - 80$
[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

I tempi indicati sono stati misurati alle seguenti tensioni:
(se non diversamente specificato)

	"1"	"0"
CLOCK	V_{CC}-0.6V	0.45V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	±0.5V

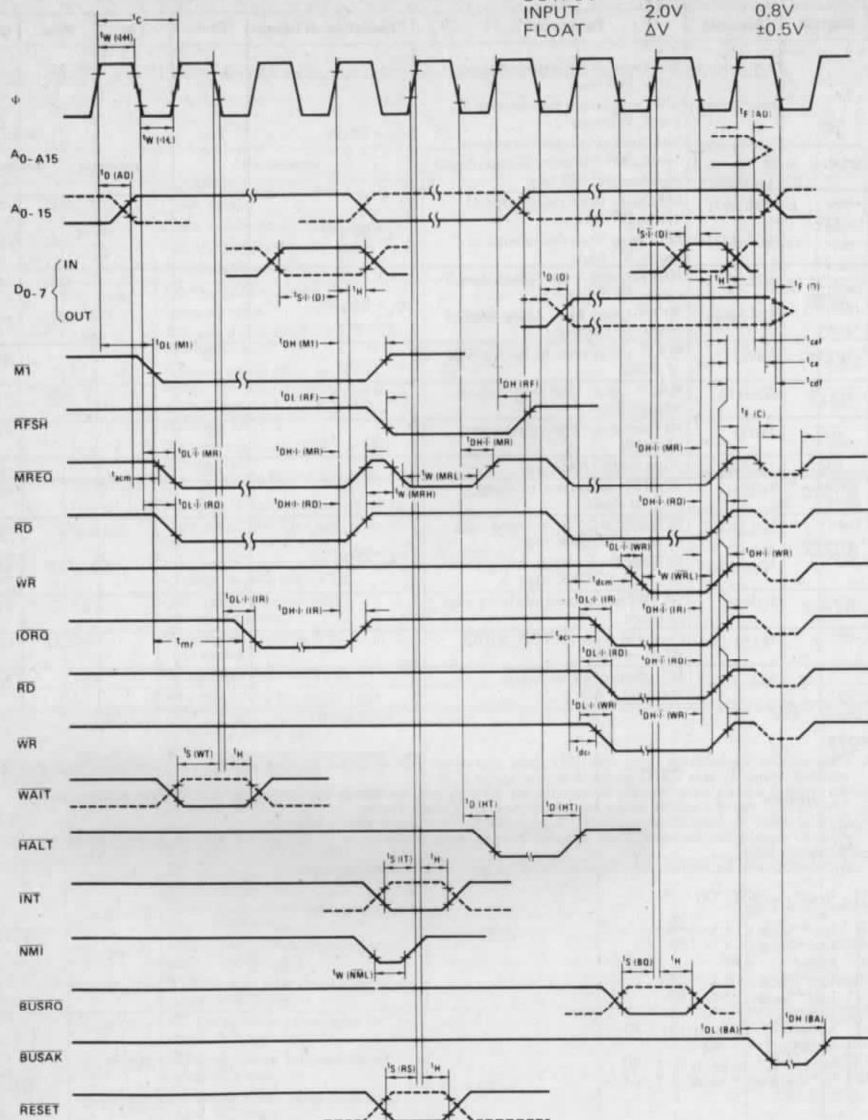


Figura 1-10. CPU Z80 diagramma delle temporizzazioni.

perchè il dispositivo di I/O sente la presente contemporanea di $\overline{BWR}$ e $\overline{BIORQ}$ attivi e del suo numero, 01, sul bus degli indirizzi, e "catturerà" immediatamente i dati dal bus dei dati.

Si noti che viene automaticamente inserito uno stato di attesa T_W; viene così dato al dispositivo periferico un po' più di tempo per rispondere anche se tale ritardo non è stato richiesto dal dispositivo stesso e la linea $\overline{BWAIT}$ é rimasta perciò alta (1). Le temporizzazioni descritte possono essere ritrovate nello Z80 Microcomputer System pubblicato dalla SGS-ATES e nel Manuale Tècnico della CPU. Siete vivamente invitati a studiare attentamente queste informazioni in modo da capirne l'interpretazione e le specifiche.

Questo capitolo vi ha rapidamente introdotto in tutti gli aspetti base riguardanti l'interfacciamento dello Z80. Ognuno degli argomenti qui presentati verrà ampliato ed esaminato nei particolari nei capitoli seguenti, sia attraverso la discussione di esperimenti illustrativi nei quali realizzerete circuiti di interfaccia, sia attraverso una analisi degli schemi del Nanocomputer. Con questo capitolo speriamo di aver creato in voi un interesse per l'interfacciamento e, senza avervi costretto ad affrontare troppe cose in una volta sola, ci auguriamo di avere risvegliato la vostra curiosità sull'arte e sulla scienza dell'interfacciamento.

CAPITOLO 2

LA SCHEDA PER ESPERIMENTI NEZ80 DELL'NBZ80-S

INTRODUZIONE

Questo capitolo contiene la descrizione della scheda per esperimenti NEZ80 appositamente costruita dall'SGS-ATES per essere impiegata unitamente al vostro Nanocomputer Super (NBZ80-S) per eseguire gli esperimenti di interfacciamento con i microcomputer. E' compresa una serie completa di schemi affinchè possiate formarvi una conoscenza completa dei circuiti impiegati per gli indicatori di monitor, gli interruttori a logica antirimbalzo, i pulser senza rimbalzi e quanti altri componenti si trovano associati alla scheda stessa.

OBIETTIVI

Al termine di questo capitolo sarete in grado di.

- Leggere correttamente gli schemi relativi alla scheda per esperimenti NEZ80.
- Condurre alcuni esperimenti di elementare difficoltà legati alla funzione degli interruttori, degli indicatori di monitor e dei pulser.
- Conoscere, almeno sommariamente, la metà circa dei segnali riportati in uscita sui tre connettori a 40 pin presenti sulla scheda per montaggi. Per i segnali restanti si procederà ad un esame approfondito nei capitoli successivi.

GENERALITA'

In Figura 2-1 è riprodotta una vista superiore della scheda per esperimenti NEZ80. I componenti sono:

I connettori J1, J2 e J3
Un connettore PIO a 40 pin
Una basetta di montaggio (Breadboard)
I connettori DIP a 40 pin A, B e C
Otto indicatori luminosi contrassegnati, da destra a sinistra
LM0, LM1, LM2, LM3, LM4, LM5, LM6 e LM7
Otto interruttori logici contrassegnati, da destra a sinistra
SW0, SW1, SW2, SW3, SW4, SW5, SW6 ed SW7
Due pulser contrassegnati P0 (a destra) e P1 (a sinistra)
Una spia d'alimentazione contrassegnata +5V.

Nei paragrafi che seguono si procede, per l'appunto, alla descrizione particolareggiata di ciascuno dei componenti sopraelencati.

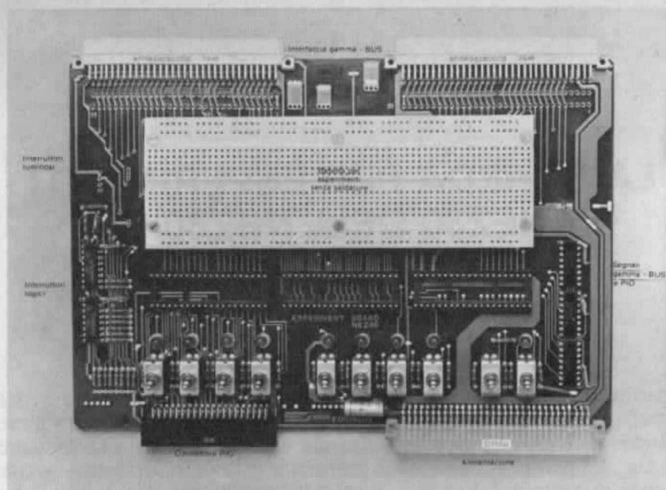
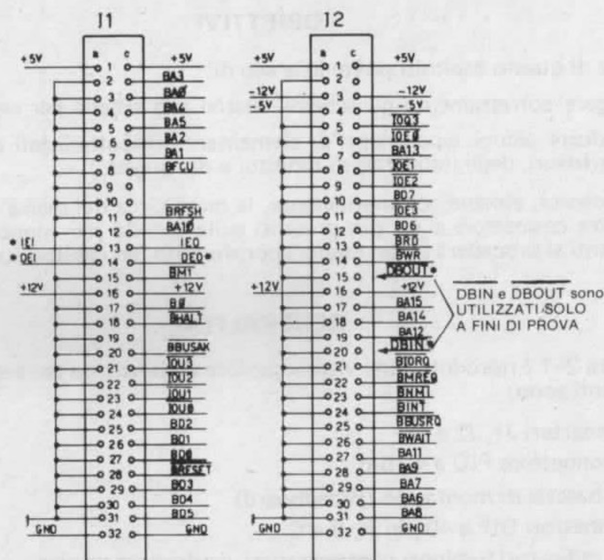


Figura 2-1. Scheda per esperimenti NEZ80 – Vista dall'alto.



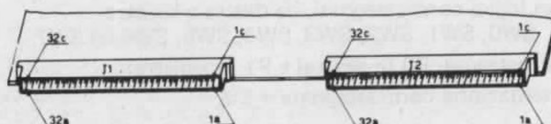


Figura 2-2. Connettori J1 e J2.

Connettori J1, J2 e J3

I connettori J1 e J2 costituiscono una delle parti più importanti dell'interfaccia tra la scheda base del Nanocomputer e la scheda per esperimenti. Grazie a questi connettori i segnali presenti sul bus del Nanocomputer possono essere scambiati tra i circuiti appartenenti alla scheda base ed i circuiti da voi montati valendovi della basetta di montaggio e dei tre zoccoli a 40 pin A, B e C. Quanto alla configurazione dei pin dei connettori J1 e J2 riportata in Figura 2-2, lo studente non deve preoccuparsi se non è in grado di comprendere i nomi assegnati ai singoli pin, in quanto di essi si avrà occasione di parlare in seguito.

Nella sezione che descrive gli zoccoli A, B e C a 40 pin si noterà come quasi tutti i segnali di J1 e J2 sono riportati in uscita sugli zoccoli A, B e C. I circuiti da voi montati si interfacciano con i circuiti della scheda della CPU mediante fili di collegamento con gli zoccoli a 40 pin.

Il connettore J3 è adibito all'alimentazione della scheda di montaggio. Come potrete notare nella descrizione dei pin degli zoccoli A, B e C, sono disponibili sulla scheda tutte le seguenti tensioni: +5V, -5V, +12V e -12V. La rappresentazione schematica di J3 è riportata in Figura 2-3.

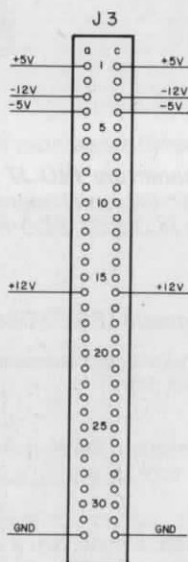


Figura 2-3. Connettore J3.

Il connettore PIO

La scheda base del Nanocomputer comprende due chip di I/O parallelo Z80-PIO. Il PIO N. 1 è adibito alle comunicazioni con l'unità tastiera/display del Nanocomputer. Il PIO N. 2 è a libera disposizione dell'utente che ne conosca l'uso. L'utilizzazione del PIO N. 2 presuppone la presenza di determinati segnali. Il connettore PIO interfaccia la scheda di montaggio con il PIO N. 2 tramite un caso flessibile a 40 poli,

collegando in tal modo vari pin appartenenti al chip PIO ai fori degli zoccoli DIP a 40 pin A, B e C. I fori adibiti a questo collegamento sono illustrati nella sezione dedicata agli zoccoli A, B e C. In Figura 2-4 si può esaminare lo schema del connettore PIO, che si collega con il connettore J7 sulla scheda della CPU (NBZ80).
I segnali associati a ciascuno dei nomi usati in figura, quali PCn, PDn, CRDY, $\overline{\text{CSTB}}$, ecc., sono descritti nella sezione dedicata agli zoccoli A, B e C.

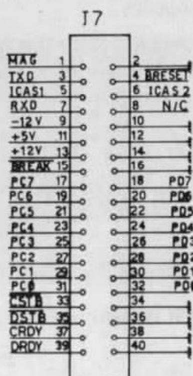


Figura 2-4. Il connettore PIO J7 della scheda base e della scheda per esperimenti. Nota: Gli unici segnali riportati sugli zoccoli A, B e C dell'NEZ80 sono PC0-PC7, PD0-PD7, $\overline{\text{CSTB}}$, $\overline{\text{DSTB}}$, CRDY e DRDY.

La basetta di montaggio (BREADBOARD)

La basetta di montaggio (breadboard) presente sulla vostra scheda per esperimenti NEZ80 è fabbricata dalla

E & L Instruments, Inc.
(rappresentata in Italia dalla Microlem di Milano o da altri costruttori)
ed è disponibile anche presso i migliori rivenditori

Il modello della E&L è noto con il nome di basetta SK-10. Inoltre zoccoli di montaggio equivalenti sono reperibili presso rivenditori di componentistica al dettaglio. La basetta è concepita in modo da fissare chip di circuiti integrati, resistori, condensatori e quanti altri componenti occorrerà usare per la realizzazione degli esperimenti descritti nel testo.

Una vista dall'alto della basetta è riportata in Figura 2-5. La basetta presenta 64 x 2 insiemi di cinque terminali a non saldare collegati elettricamente, disposti simmetricamente rispetto ad una sottile striscia centrale, ed otto insieme di venticinque terminali collegati elettricamente, disposti lungo i lati maggiori. I gruppi centrali di cinque terminali collegati elettricamente hanno la funzione di alloggiare i circuiti integrati, consentendo sino a quattro ulteriori collegamenti con ciascuno dei pin dei circuiti integrati più piccoli a 14 o 16 pin. I gruppi di venticinque terminali collegati elettri-

camente, ai bordi della piastra, sono adibiti all'alimentazione dei circuiti integrati e degli altri componenti. La faccia inferiore della basetta privata del suo rivestimento protettivo, è riprodotta in Figura 2-6. Si notino i collegamenti tra i terminali a non saldare.

La basetta è corredata di serie di una custodia stampata in plastica anti-urto e presenta in tutto 840 contatti a molla con elevata resistenza agli sforzi e anticorrosione,

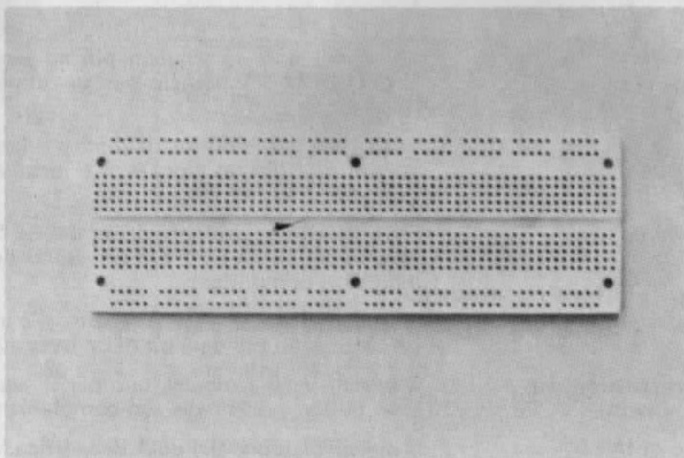


Figura 2-5. Vista dall'alto della basetta di montaggio (breadboard).

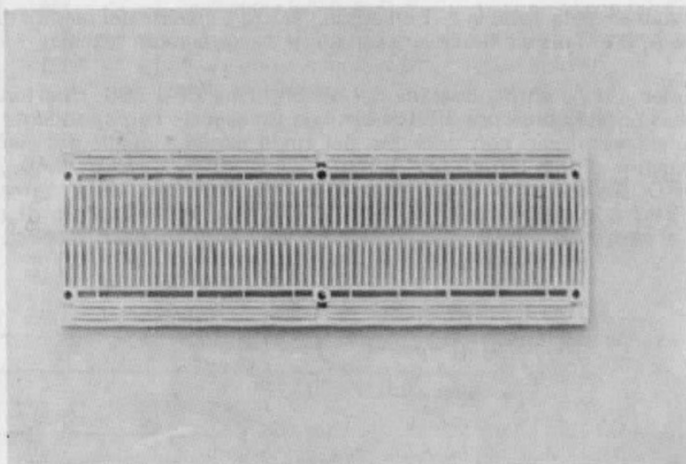


Figura 2-6. Vista dal basso della basetta di montaggio.

progettati per una vita utile pari a oltre diecimila inserzioni. I terminali a non saldare accolgono fili di misura compresa tra il n. 22 ed n. 26; la misura consigliata è il n. 24, della quale l'SGS-ATES fornisce il kit K1Z80 di fili.

I connettori A, B e C consistono in zoccoli standard per chip in contenitori a doppia fila 40 piedini, collegati a J1, J2 ed al connettore PIO tramite i circuiti della scheda di montaggio. Sugli zoccoli A, B e C sono riportati, in totale, 86 segnali diversi, ciascuno dei quali può essere utilizzato in un circuito montato sulla basetta mediante il semplice collegamento con fili tra il circuito della basetta e gli opportuni fori negli zoccoli A, B e/o C. I segnali disponibili sugli zoccoli A, B e C possono essere distinti per categorie:

CPU Z80 (Bus bufferizzato)	:	38 segnali uno per ciascun pin ad esclusione di Vcc e GND +1 segnale per un impulso $\overline{BWR}$ ritardato.
PIO N. 2	:	21 segnali per due bus dei dati e relativi segnali di controllo più un segnale di priorità di interruzione (IEO).
Indicatori luminosi	:	8 segnali di ingresso per i driver dei LED, uno per ciascun bit di un byte a 8 bit. Sono detti anche indicatori di monitor.
Interruttori	:	8 segnali di uscita degli interruttori a logica antirimbalzo, uno per ogni bit di un byte a 8 bit.
Pulser (Interruttori con ritorno pulsanti)	:	4 segnali senza rimbalzi, uno per il valore di ciascun pulser, nonchè del suo complemento logico.
Controllo di I/O	:	9 segnali ciascuno dei quali decodifica in parte gli otto bit inferiori del bus degli indirizzi.
Alimentazione	:	4 segnali: +5V, -5V, +12V e GND

Più specificatamente, la Tabella 2-1 chiarisce l'esatta funzione dei piedini di ciascuno degli zoccoli A, B e C oltre a descrivere sommariamente ciascun segnale.

Da un riesame della configurazione dei piedini della CPU Z80, riportata nel capitolo 1, si può notare come per 38 di essi vi sia un segnale corrispondente riportato sulla scheda di montaggio con etichetta del tutto simile a quella del piedino dello Z80 ad eccezione di una lettera "B" iniziale, ad es. BD0, . . . BD7, BA0, . . . BA15, $\overline{BMREQ}$, $\overline{BRD}$, $\overline{BWR}$. La "B" significa *buffered* (bufferizzato, ossia trasmesso tramite buffer). Essa è impiegata per ricordarvi che il foro nello zoccolo di montaggio è collegato al corrispondente piedino della CPU Z80 con l'interposizione di un buffer (Fig. 2-7).



Figura 2-7. I fori presenti sulla basetta sono collegati ai piedini della CPU non direttamente ma attraverso un buffer.

Tabella 2-1. Disposizione dei piedini degli zoccoli a 40 pin A, B e C.

ZOCCOLO A		
Piedino	**Segnale**	**Descrizione**
1		Non utilizzato
2		Non utilizzato
3		Non utilizzato
4		Non utilizzato
5	P1	Complemento del valore logico del pulser N. 1
6	P1	Valore logico del pulser N. 1
7		Non utilizzato
8	P0	Complemento del valore logico del pulser N. 0
9	P0	Valore logico del pulser N. 0
10		Non utilizzato
11		Non utilizzato
12	SW7	Interruttore logico N. 7
13	SW6	Interruttore logico N. 6
14	SW5	Interruttore logico N. 5
15	SW4	Interruttore logico N. 4
16	SW3	Interruttore logico N. 3
17	SW2	Interruttore logico N. 2
18	SW1	Interruttore logico N. 1
19	SW0	Interruttore logico N. 0
20		Non utilizzato
21		Non utilizzato
22	LM0	Indicatore luminoso N. 0
23	LM1	Indicatore luminoso N. 1
24	LM2	Indicatore luminoso N. 2
25	LM3	Indicatore luminoso N. 3
26	LM4	Indicatore luminoso N. 4
27	LM5	Indicatore luminoso N. 5
28	LM6	Indicatore luminoso N. 6
29	LM7	Indicatore luminoso N. 7
30		Non utilizzato
31		Non utilizzato
32	GND	Massa
33	-12V	Alimentazione a -12 volt
34	GND	Massa
35	+12V	Alimentazione a +12 volt
36	GND	Massa
37	-5V	Alimentazione a -5 volt
38	GND	Massa
39	+5V	Alimentazione a +5 volt
40	GND	Massa

ZOCCOLO B		
Piedino	**Segnale**	**Descrizione**
1	$\overline{BM1}$	Segnale $\overline{M1}$ bufferizzato in uscita dal chip CPU Z80
2	$\overline{BMREQ}$	Segnale $\overline{MREQ}$ bufferizzato in uscita dal chip CPU Z80
3	$\overline{BIORQ}$	Segnale $\overline{IORQ}$ bufferizzato in uscita dal chip CPU Z80
4	$\overline{BRFSH}$	Segnale $\overline{RFSH}$ bufferizzato in uscita dal chip CPU Z80
5		Non utilizzato
6	$\overline{BRD}$	Segnale $\overline{RD}$ bufferizzato in uscita dal chip CPU Z80
7	$\overline{BWR}$	Segnale $\overline{WR}$ bufferizzato in uscita dal chip CPU Z80
8	$\overline{DBWR}$	Segnale $\overline{BWR}$ ritardato di 100 nsec
9		Non utilizzato

Tabella 2-1. Continuazione

ZOCCOLO B		
Piedino	**Segnale**	**Descrizione**
10	$\overline{\text{BHALT}}$	Segnale $\overline{\text{HALT}}$ bufferizzato in uscita dal chip CPU Z80
11	$\overline{\text{BWAIT}}$	Segnale $\overline{\text{WAIT}}$ bufferizzato in ingresso dal chip CPU Z80 (Deve essere un'uscita con collettore aperto del circuito dello sperimentatore)
12	$\overline{\text{BINT}}$	Segnale $\overline{\text{INT}}$ bufferizzato in ingresso al chip CPU Z80 (Deve essere un'uscita con collettore aperto del circuito dello sperimentatore)
13	$\overline{\text{BNMI}}$	Segnale $\overline{\text{NMI}}$ bufferizzato in ingresso al chip CPU Z80 (Deve essere un'uscita con collettore aperto del circuito dello sperimentatore)
14	$\overline{\text{BRESET}}$	Segnale $\overline{\text{RESET}}$ bufferizzato in ingresso al chip CPU Z80 (Deve essere un'uscita con collettore aperto del circuito dello sperimentatore)
15		Non utilizzato
16	$\overline{\text{BBUSRQ}}$	Segnale $\overline{\text{BUSRQ}}$ bufferizzato in ingresso al chip CPU Z80 (Deve essere un'uscita con collettore aperto del circuito dello sperimentatore)
17	$\overline{\text{BBUSAK}}$	Segnale $\overline{\text{BUSAK}}$ bufferizzato in uscita dal chip CPU Z80
18		Non utilizzato
19	$\overline{\text{B}\Phi}$	Clock bufferizzato dello Z80
20		Non utilizzato
21		Non utilizzato
22	BD0	Linea D0 bufferizzata del bus dei dati bidirezionale dello Z80
23	BD1	Linea D1 bufferizzata del bus dei dati bidirezionale dello Z80
24	BD2	Linea D2 bufferizzata del bus dei dati bidirezionale dello Z80
25	BD3	Linea D3 bufferizzata del bus dei dati bidirezionale dello Z80
26	BD4	Linea D4 bufferizzata del bus dei dati bidirezionale dello Z80
27	BD5	Linea D5 bufferizzata del bus dei dati bidirezionale dello Z80
28	BD6	Linea D6 bufferizzata del bus dei dati bidirezionale dello Z80
29	BD7	Linea D7 bufferizzata del bus dei dati bidirezionale dello Z80
30		Non utilizzato
31	$\overline{\text{IOQ3}}$	Impulso negativo generato nel caso che A7-A0 sia uguale a 0000 11xx
32	$\overline{\text{IOE0}}$	Impulso negativo generato nel caso che A7-A0 sia uguale a 0001 00xx
33	$\overline{\text{IOE1}}$	Impulso negativo generato nel caso che A7-A0 sia uguale a 0001 01xx
34	$\overline{\text{IOE2}}$	Impulso negativo generato nel caso che A7-A0 sia uguale a 0001 10xx
35	$\overline{\text{IOE3}}$	Impulso negativo generato nel caso che A7-A0 sia uguale a 0001 11xx
36	$\overline{\text{IOU0}}$	Impulso negativo generato nel caso che A1-A0 sia uguale a 00 e $\overline{\text{BIORQ}}$ sia attivo (0 logico)
37	$\overline{\text{IOU1}}$	Impulso negativo generato nel caso che A1-A0 sia uguale a 01 e $\overline{\text{BIORQ}}$ sia attivo (0 logico)
38	$\overline{\text{IOU2}}$	Impulso negativo generato nel caso che A1-A0 sia uguale a 10 e $\overline{\text{BIORQ}}$ sia attivo (0 logico)
39	$\overline{\text{IOU3}}$	Impulso negativo generato nel caso che A1-A0 sia uguale a 11 e $\overline{\text{BIORQ}}$ sia attivo (0 logico)
40		Non utilizzato

Si osservi che i piedini 31-39 sono tutti associati a segnali rappresentanti una decodifica parziale del bus degli indirizzi, potendo quindi essere utilizzati in coppia in modo da dar luogo ad un unico impulso di selezione dell'indirizzo di I/O.

Tabella 2-1. Continuazione

ZOCCOLO C		
Piedino	**Segnale**	**Descrizione**
1	CRDY	Segnale di Porta A pronta da PIO N. 2
2	$\overline{CSTB}$	Segnale di strobe della porta A inviato in ingresso a PIO N. 2
3	PC7	Linea D7 del bus dei dati bidirezionale della Porta A per PIO N. 2
4	PC6	Linea D6 del bus dei dati bidirezionale della Porta A per PIO N. 2
5	PC5	Linea D5 del bus dei dati bidirezionale della Porta A per PIO N. 2
6	PC4	Linea D4 del bus dei dati bidirezionale della Porta A per PIO N. 2
7	PC3	Linea D3 del bus dei dati bidirezionale della Porta A per PIO N. 2
8	PC2	Linea D2 del bus dei dati bidirezionale della Porta A per PIO N. 2
9	PC1	Linea D1 del bus dei dati bidirezionale della Porta A per PIO N. 2
10	PC0	Linea D0 del bus dei dati bidirezionale della Porta A per PIO N. 2
11	DRDY	Segnale di Porta B pronta da PIO N. 2
12	$\overline{DSTB}$	Impulso di strobe della Porta B inviato in ingresso a PIO N. 2
13	PD7	Linea D7 del bus dei dati bidirezionale della Porta B per PIO N. 2
14	PD6	Linea D6 del bus dei dati bidirezionale della Porta B per PIO N. 2
15	PD5	Linea D5 del bus dei dati bidirezionale della Porta B per PIO N. 2
16	PD4	Linea D4 del bus dei dati bidirezionale della Porta B per PIO N. 2
17	PD3	Linea D3 del bus dei dati bidirezionale della Porta B per PIO N. 2
18	PD2	Linea D2 del bus dei dati bidirezionale della Porta B per PIO N. 2
19	PD1	Linea D1 del bus dei dati bidirezionale della Porta B per PIO N. 2
20	PD0	Linea D0 del bus dei dati bidirezionale della Porta B per PIO N. 2
21		Non utilizzato
22	BA0	Linea A0 bufferizzata del bus degli indirizzi dello Z80
23	BA1	Linea A1 bufferizzata del bus degli indirizzi dello Z80
24	BA2	Linea A2 bufferizzata del bus degli indirizzi dello Z80
25	BA3	Linea A3 bufferizzata del bus degli indirizzi dello Z80
26	BA4	Linea A4 bufferizzata del bus degli indirizzi dello Z80
27	BA5	Linea A5 bufferizzata del bus degli indirizzi dello Z80
28	BA6	Linea A6 bufferizzata del bus degli indirizzi dello Z80
29	BA7	Linea A7 bufferizzata del bus degli indirizzi dello Z80
30		Non utilizzato
31	BA8	Linea A8 bufferizzata del bus degli indirizzi dello Z80
32	BA9	Linea A9 bufferizzata del bus degli indirizzi dello Z80
33	BA10	Linea A10 bufferizzata del bus degli indirizzi dello Z80
34	BA11	Linea A11 bufferizzata del bus degli indirizzi dello Z80
35	BA12	Linea A12 bufferizzata del bus degli indirizzi dello Z80
36	BA13	Linea A13 bufferizzata del bus degli indirizzi dello Z80
37	BA14	Linea A14 bufferizzata del bus degli indirizzi dello Z80
38	BA15	Linea A15 bufferizzata del bus degli indirizzi dello Z80
39		Non utilizzato
40		Non utilizzato

Relativamente a PIO N. 2 si ricordano i seguenti indirizzi: porta C (dati) = 08H, porta C (controllo) = 0AH, porta D (dati) = 09, porta D (controllo) = 0BH.

Una conseguenza importante di questa struttura a buffer sui segnali da e verso la CPU Z80 è la protezione della CPU stessa. Anche il più meticoloso degli sperimentatori può, infatti, commettere degli errori occasionali nella disposizione dei collegamenti circuitali sulla basetta, a loro volta collegati alla CPU attraverso gli zoccoli A, B e C. L'interposizione di buffer per i segnali che interessano la CPU, permette un

piccolo ma utile margine di errore. L'interposizione di buffer non isola completamente il circuito sulla basetta dalla CPU, per cui occorre prestare sempre la massima attenzione, ma può essere determinante per farvi evitare la perdita di alcuni chip.

Oltre ad assolvere questa funzione protettiva, le uscite dei buffer conferiscono ai segnali presenti sulla piastra di montaggio la capacità di pilotare 15 carichi TTL oppure 60 carichi Schottky low-power, purchè sia attivo il relativo segnale di abilitazione. In conseguenza a questa superiore capacità di pilotaggio, quasi tutti i circuiti di microcalcolatori prevedono una struttura a buffer per i segnali della CPU.

Se da un lato la trasmissione dei segnali via buffer non cambia obbligatoriamente il loro stato logico, può d'altronde succedere che cambi in modo significativo la relazione temporale dell'interazione di un segnale con gli altri. Come si vedrà meglio nei prossimi capitoli, è assai importante distinguere tra i segnali della CPU e quelli bufferizzati ad essi associati, presenti sul bus e perciò nel seguito ci si impegnerà scrupolosamente nel conservare sempre queste notazioni convenzionali:

I segnali relativi alla CPU hanno sempre un contrassegno privo di una "B" iniziale.
Nei casi in cui dei circuiti analoghi a quello del Nanocomputer, sono presentati servendosi direttamente dei segnali della CPU, questi sono da intendersi unicamente come esempi, il cui funzionamento sarebbe corretto a condizione che sul bus fossero presenti i segnali della CPU. E' evidente che questo non è certo il caso del Nanocomputer, né, peraltro, della maggior parte degli altri sistemi reali di microcalcolatori. Ogniqualvolta, a proposito di un esperimento, sarà presentato uno schema, ci si varrà dei segnali del bus, che saranno, perciò, contrassegnati da una "B" iniziale. In tutti gli schemi di montaggio ci si varrà di etichette in accordo con quelle riportate sulla piastra di montaggio.

La maggior parte dei segnali sopraelencati sono portati sugli zoccoli A, B e C mediante collegamento diretto con i pin di J1, J2 o del connettore PIO. Fa eccezione il segnale $\overline{DBWR}$, che non è altro che il segnale $\overline{BWR}$ ritardato di 100 nanosecondi. Il circuito utilizzato per la generazione di $\overline{DBWR}$ è riportato in Figura 2-8.

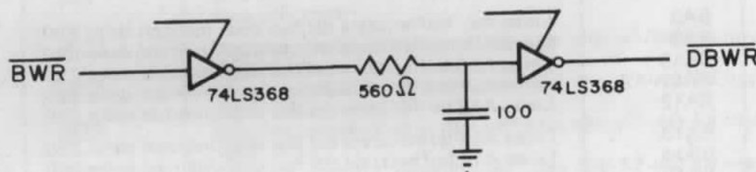


Figura 2-8. Il circuito per $\overline{DBWR}$.

Il lettore si ricordi di un consiglio già dato in queste pagine: non preoccupatevi di non riconoscere o comprendere molti tra i segnali descritti nelle pagine precedenti. Queste tabelle sono fornite soprattutto a scopo di riferimento e di rassegna del tipo e del numero di segnali a vostra disposizione per l'interfacciamento del microcalcolatore. Nel corso di questo capitolo, sarete invitati ad eseguire alcuni semplici esperimenti che prevedono l'impiego di indicatori di monitor, switch, pulser, e dei livelli di tensione dell'alimentazione GND e +5V. Degli altri segnali si tratterà particolareggiatamente nei prossimi capitoli.

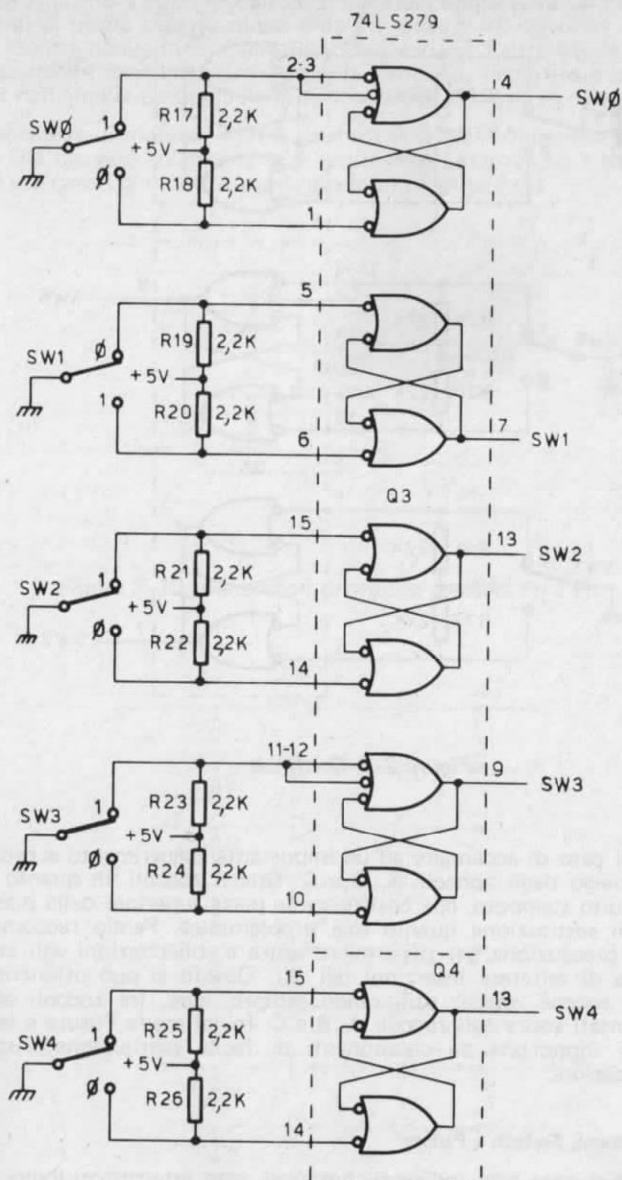


Figura 2-9. Gli interruttori logici SWn, dove n = 0, . . ., 7.

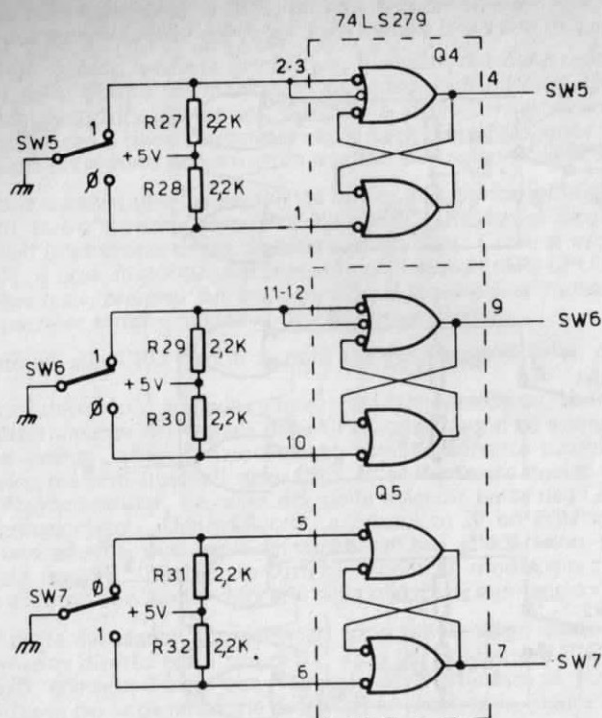


Figura 2-9. Continua

Ci sembra il caso di accennare ad un importante suggerimento a proposito delle modalità d'impiego degli zoccoli A, B e C. Questi zoccoli, in quanto saldati alla scheda del circuito stampato, che costituisce la parte superiore della piastra di montaggio, sono di sostituzione quanto mai problematica. Perciò raccomandiamo di prendere ogni precauzione per risparmiare usura e sollecitazioni agli zoccoli stessi in conseguenza di reiterare inserzioni dei fili. Questo si può ottenere ricorrendo all'impiego di *zoccoli doppi*, utilizzando sempre, cioè, tre zoccoli addizionali a 40 piedini montati sopra agli zoccoli A, B e C. In tal modo l'usura e le varie sollecitazioni sono sopportate da componenti di facile sostituzione, risparmiandovi spese e complicazioni.

Indicatori luminosi, Switch e Pulser

Sulla piastra ci sono otto indicatori luminosi, otto interruttori logici e due interruttori con molla di ritorno con funzione di dispositivi di ingresso (switch e pulser) e di uscita (indicatori luminosi). Tanto gli interruttori che i pulser logico sono tutti antirimbalzo, valendosi di circuiti che impiegano due porte NAND. Gli schemi relativi agli switch ed ai pulser sono riportati nelle Figure 2-9 e 2-10.

Si osservi come ogni switch determini uno stato di 0 logico oppure di 1 logico al proprio terminale di uscita, costituito da un foro nello zoccolo A contrassegnato SWn, dove n = 0, 1, . . ., o 7. Ogni pulser determina la comparsa di un impulso isolato ai due terminali di uscita contrassegnati Pn e $\overline{Pn}$, dove n = 0 oppure 1. L'impulso in Pn è positivo, mentre quello in $\overline{Pn}$ è negativo. Sia l'uno che l'altro dei due Pulser è attivato dallo spostamento dello switch ad esso associato, provvisto a sua volta di una molla per il richiamo automatico nella sua posizione normale di riposo.

Ogni indicatore di monitor è OFF (spento) se il relativo ingresso è a massa oppure floating, e ON (acceso) solamente se il suo ingresso si trova ad 1 logico. Gli schemi relativi agli indicatori di monitor sono riportati in Figura 2-11.

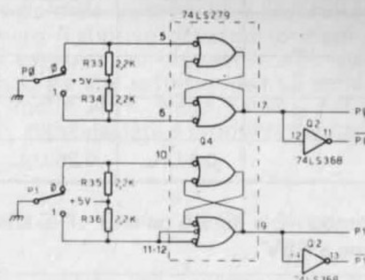


Figura 2-10. Generatori di impulsi (pulser) Pn e $\overline{Pn}$.

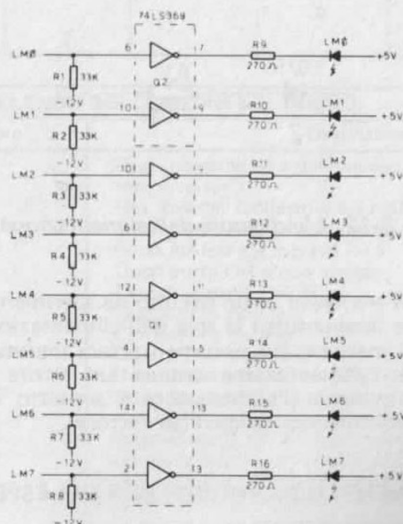


Figura 2-11. Gli indicatori luminosi LMn, dove n = 0, . . ., 7.

La Tabella 2-2 illustra il carico presentato al circuito sulla basetta degli indicatori luminosi e il carico che può essere pilotato sulla basetta degli interruttori logici e dai pulsanti.

Tabella 2-2. Fan-In e Fan-Out per interruttori logici, generatori di impulsi (pulser) e indicatori luminosi.

Componenti sul Breadboard		FAN-IN (Carico)		FAN-OUT (Pilota)	
		Alto	Basso	Alto	Basso
Interruttori Logici				10 U.L.	5 U.L.
Generatori di impulsi	Pn			10 U.L.	5 U.L.
	$\overline{Pn}$			85 U.L.	15 U.L.
Indicatori Luminosi		0,5 U.L.	0,25 U.L.		

Indicatore dell'alimentazione a +5V

L'indicatore dell'alimentazione a +5V consiste in un semplice diodo ad emissione luminosa (LED = Light Emitting Diode) che si accende ogni qualvolta si applica l'alimentazione di +5V alla scheda di montaggio. Il relativo circuito compare in Figura 2-12.

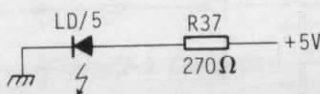


Figura 2-12. L'indicatore dell'alimentazione a +5V.

Tutte le volte che vi troverete in un circuito da sperimentare che non funziona ricordatevi di controllare innanzitutto la spia dell'alimentazione. Vi sorprenderete del numero di volte che il mancato funzionamento sarà imputabile al fatto che vi siete dimenticati di fornire l'alimentazione oppure un errore nei collegamenti ne ha determinato il corto circuito (l'alimentatore è protetto contro questo incidente e la tensione si ristabilirà non appena corretto l'errore).

COMPONENTI OCCORRENTI PER GLI ESPERIMENTI

In questa sezione presentiamo nella Tavola 2-1, un elenco di tutti i componenti, compresi i fili, a voi occorrenti per eseguire tutti gli esperimenti descritti nel testo. Presso l'SGS-ATES sono reperibili i Kit per esperimenti, K1Z80 (fili) e K2Z80 (componenti), che contengono tali componenti al completo.

LA FAMIGLIA DEI DISPOSITIVI TTL E LE FAMIGLIE DA ESSA DERIVATE

Come si può notare nell'elenco dei componenti riportato nella Tavola 2-1, i dispositivi TTL 7400 sono utilizzati negli esperimenti descritti nel testo, appartengono tutti quanti alla *sottofamiglia TTL: low-power Schottky* (low power Schottky = LS), TTL sta per transistor-transistor logic (logica transistore-transistore), circuiti integrati che sono stati progettati in modo da poter essere combinati sino a realizzare circuiti logici di complessità superiore. La maggioranza dei fabbricanti di circuiti integrati TTL seguono il sistema di numerazione 7400. Tutti i dispositivi presentano, cioè, un numero di catalogo del tipo:

74XXNNN

dove XX rappresenta la sigla della sottofamiglia costituita da zero, una, oppure due lettere, ed NNN un numero distintivo del chip formato da due o tre cifre. Alcuni esempi di tali numeri di catalogo dei dispositivi TTL compaiono nella Tabella dei componenti, Tabella 2-2. Oltre alla sottofamiglia LS della serie TTL, di cui abbiamo già parlato, ve ne sono molte altre. Nella Tabella 2-3 è riportata, appunto, una panoramica dei loro nomi, abbreviazioni e proprietà fondamentali.

Tabella 2-1. Elenco dei fili e dei componenti.

ELENCO DEI FILI (Misura n. 24) (K1Z80)

Colore	10"	6"	4"	2"	1"
Verde	4	12	8	8	
Giallo	4	12	8	8	
Nero			8	8	10
Rosso			8	8	10
Azzurro	2				

ELENCO DEI COMPONENTI (K2Z80)

Quantità	Dispositivo	Descrizione
1	74LS02	Quad porta NOR a due ingressi
1	74LS04	Hex inverter
1	74LS05	Hex inverter (collettore aperto)
1	74LS08	Quad porta AND a due ingressi
1	74LS30	Porta NAND a 8 ingressi
1	74LS32	Quad porta OR a due ingressi
1	74LS42	Decodificatore 1 su 10
1	74LS74	Doppio flip-flop D
1	74LS90	Contatore a decade
1	74LS125	Quad buffer three-state (abilitazione livello basso)
1	74LS139	Doppio Decoder demultiplexer 1 su 4
2	74LS175	Quad flip-flop D con azzeramento
2	74LS365	Hex buffer con abilitazione comune (three-state)
1	Z80-PIO	Dispositivo di I/O per interfaccia parallela CPU Z80
1	Z80-CTC	Dispositivo contatore/temporizzatore per CPU Z80
1	555	Oscillatore astabile/multivibratore monostabile
2	2101A	RAM statica 4 x 256
3	LED rosso	Diodo ad emissione luminosa rosso
3	330 ohm, 1/4 watt	Resistenza
6	1 Kohm, 1/4 watt	Resistenza
1	3 3 Kohm, 1/4 watt	Resistenza
1	1000 pF	Condensatore

Tabella 2-3. Sottofamiglie TTL — Panoramica generale

Nome	Sigla della sottofamiglia (xx)	Tempo di propagazione di Gate	Consumo di un Gate	Massima frequenza di conteggio	Osservazioni
TTL normale	—	10 ns	10 mW	35 MHz	Grande varietà Prezzo minimo Massima disponibilità
High-Power TTL (TTL di potenza)	H	6 ns	22 mW	50 MHz	E' in corso la sua sostituzione con la più recente TTL Schottky, rispetto ad essa più veloce e caratterizzata da minor consumo.
Low-Power TTL (TTL a basso consumo)	L	33 ns	1 mW	3 MHz	E' in corso la sua sostituzione con le famiglie logiche CMOS
TTL Schottky	S	3 ns	19 mW	125 MHz	Migliore rapporto velocità/consumo rispetto ai TTL normali
Low-Power Schottky TTL	LS	10 ns	2 mW	45 MHz	Rapporto velocità/consumo migliore rispetto alle TTL tradizionali. Nonostante il suo carattere di novità e la maggiore complessità è tuttavia da preferirsi, grazie anche a miglioramenti come disponibilità, varietà, prezzo e consumo.

Tutti gli elementi che hanno lo stesso numero NNN sono assolutamente uguali *dal punto di vista logico.* Tutti i CI che seguono sono, per esempio, quad porte NAND a due ingressi con identica configurazione dei pin: 7400, 74L00, 74H00, 74S00 e 74LS00. Si badi, tuttavia, che in alcuni casi la configurazione dei piedini non è identica. I chip 7485, 74L85 e 74LS85 fabbricati dalla Texas Instruments Inc. non presentano, ad esempio la stessa configurazione. Sorge perciò naturale la domanda: se per un esperimento del testo è richiesto un chip 74LS ed io possiedo un chip 74, 74L, 74H, 74S o 74LS dello stesso numero, mi è consentito utilizzarlo al posto di quello nominato? Malauguratamente, la risposta a questa domanda dipende dalla situazione, in quanto è necessario tener conto del FAN-IN e del FAN-OUT dei singoli chip e del circuito. La definizione di FAN-IN e FAN-OUT è la seguente:

FAN-IN Fan-in rappresenta il carico di ingresso di un ingresso digitale di circuito integrato. Per la famiglia logica TTL, il carico di ingresso è normalizzato con riferimento al valore di 1 unità di Carico TTL (TTL Unit Load = U.L. = carico unitario), pari a 1,6 mA nello stato di 0 logico e 0,04 mA nello stato di 1 logico, all'ingresso.

FAN-OUT Fan-out rappresenta la capacità di pilotaggio di un'uscita digitale di circuito integrato. Per la famiglia logica TTL, la capacità di pilotaggio dell'uscita ha un valore di 10 U.L., ovvero 16 mA, nello stato di 0 logico e 0,4 mA nello stato logico 1.

Esempi

1. Una porta 7400, avente un carico massimo di ingresso nello stato basso (denominato I_{IL}) pari a 1,6 mA ed un carico massimo di ingresso nello stato alto (denominato I_{IH}) pari a 0,04 mA, presenta, per definizione, un FAN-IN UNITARIO.

2. Per la porta 74LS00: I_{IL} = 0,36 mA e I_{IH} = 0,02 mA. Il 74LS00 presenta, perciò, un *fattore di carico basso* di

$$\frac{0{,}36\ \text{mA}}{1{,}6\ \text{mA}}$$

ossia 0,225 U.L., ed un *fattore di carico alto* di

$$\frac{0{,}02\ \text{mA}}{0{,}04\ \text{mA}}$$

ossia 0,5 U.L.

3. L'uscita del 7400 può assorbire (sink) 16 mA nello stato basso e costituisce una sorgente di 0,8 mA nello stato alto. Perciò *il fattore di pilotaggio basso* in uscita vale

$$\frac{16\ \text{mA}}{1{,}6\ \text{mA}} = 10\ \text{U.L.}$$

Il *fattore di pilotaggio alto* in uscita vale

$$\frac{0{,}8\ \text{mA}}{0{,}04\ \text{mA}} = 20\ \text{U.L.}$$

4. L'uscita del 74LS00 permette un assorbimento di 8,0 mA nello stato *basso* e costituisce una sorgente di 0,4 mA nello stato *alto*. Perciò *il fattore di pilotaggio basso* in uscita vale 5 U.L. ed il *fattore di pilotaggio alto* in uscita è pari a 10 U.L.

Il carico relativo ed i fattori di pilotaggio per le famiglie TTL fondamentali sono riportati nella tabella 2-4.

Tabella 2-4. Fattori di carico e di pilotaggio per famiglie TTL fondamentali

Famiglia	CARICO DI INGRESSO		CARICO DI USCITA	
	Alto	Basso	Alto	Basso
74	1,0 U.L.	1,0 U.L.	20 U.L.	10,0 U.L.
74H	1,25 U.L.	1,25 U.L.	25 U.L.	12,5 U.L.
74S	1,25 U.L.	1,25 U.L.	25 U.L.	12,5 U.L.
74LS	0,5 U.L.	0,25 U.L.	10 U.L.	5,0 U.L.

Nota: I valori relativi ai dispositivi MSI (medium scale integration = integrazione su media scala) sono soggetti a variazioni significative da un elemento all'altro. Per le caratteristiche effettive si consultino le note tecniche relative a quel componente. Tutti i dispositivi considerati nel testo sono illustrati nei normali manuali tecnici dei circuiti integrati (CI o all'inglese, IC) TTL. In sintesi, il modo migliore per stabilire se un esperimento permette la sostituzione con un chip non LS è quello di verificare, tanto nello stato di 0 logico (basso) che in quello di 1 logico (alto), che:

1. Al suo ingresso sia disponibile una sufficiente corrente di pilotaggio dell'ingresso (fornita da altri dispositivi del circuito).
2. Esso sia in grado di fornire in uscita una sufficiente corrente di pilotaggio ai chip del circuito collegati alla sua uscita.

Se l'elemento sostitutivo in esame supera queste due prove, il suo impiego è del tutto consentito, senza perciò dar luogo ad inconvenienti.

E' d'altronde evidente che l'alternativa più semplice è quella di avere sottomano tutti i componenti necessari. I chip LS, inoltre, sono un po' più veloci dei normali chip TTL e consumano di meno.

INTRODUZIONE AGLI ESPERIMENTI

Vengono forniti due esperimenti allo scopo di farvi acquistare dimestichezza con l'uso degli interruttori logici, indicatori luminosi e pulser presenti sulla scheda di montaggio posta sul Nanocomputer. Una parte del primo esperimento è inoltre dedicata ad insegnarvi il modo corretto per alimentare la basetta.

Insistiamo ancora sull'opportunità di utilizzare degli zoccoli doppi per i collegamenti con i segnali presenti agli zoccoli A, B e C. Secondo la nostra esperienza, il filo della misura n. 24 è l'ideale per essere impiegato con una basetta. Un filo di dimensioni appena più piccole, della misura n. 22, può tuttavia essere tollerato, mentre è decisamente sconsigliabile qualsiasi filo più grosso. Non si dimentichi, tuttavia, che il filo n. 22 darà luogo ad un'usura degli zoccoli A, B e C molto più rapida che non il filo n. 24.

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è illustrare l'impiego degli interruttori logici, gli indicatori luminosi e i pulser appartenenti alla scheda per esperimenti del Nanocomputer. Inoltre si descrive il procedimento per alimentare la basetta di montaggio.

Schema del circuito

SW0 ———————————— LM0
SW1 ———————————— LM1
SW2 ———————————— LM2
SW3 ———————————— LM3
P0 ———————————— LM6
$\overline{P0}$ ———————————— LM7

Passo 1

Il primo passo (anche se non ricordato esplicitamente) per tutti gli esperimenti qui descritti consiste nell'accertarsi che la basetta di montaggio possa essere alimentata correttamente. In Figura 2-13 sono evidenziati tutti i collegamenti da effettuarsi per alimentare la basetta.

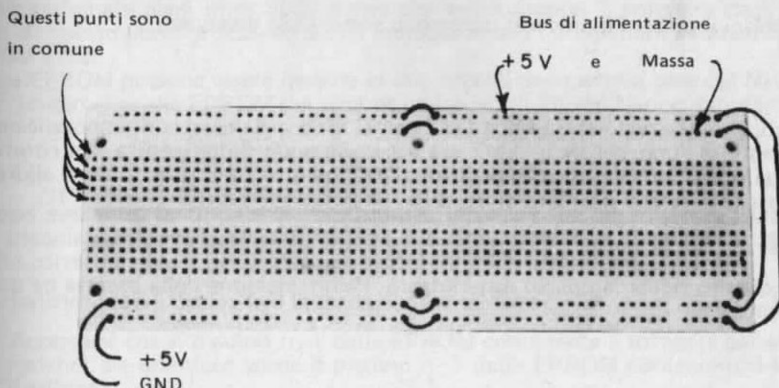


Figura 2-13. Applicazione dell'alimentazione alla basetta di montaggio.

Tre coppie di ponticelli hanno la funzione di collegare insieme tutti i terminali sui due bordi laterali della piastra, un "bus" di massa ed un "bus" a +5V. Il potenziale dei due bus è stabilito dal collegamento con i piedini di alimentazione dello zoccolo A. Il bus interno è collegato ad un pin con l'etichetta GND (abbreviazione di GROUND = massa) mentre quello esterno è collegato allo zoccolo A, piedino 39, con etichetta +5V. Si consiglia vivamente di seguire la convenzione di far corrispondere il filo rosso a +5V e quello nero a GND. Si effettuino i collegamenti di Figura 2-13 con impiego di filo nero e rosso.

Passo 2

Si proceda all'allestimento del circuito illustrato nello schema all'inizio di questo esperimento. Per far ciò collegare SW0 a LM0, SW1 a LM1 e così di seguito. Si osservi che ciò comporta l'inserzione dei fili nel solo zoccolo A, mentre con la basetta non deve essere realizzato alcun collegamento. Per i collegamenti relativi ai segnali si consiglia l'impiego alternato di filo Verde e Giallo – SW0 (verde), SW1 (giallo), ecc.

Passo 3

Con gli interruttori logici tutti in posizione OFF, corrispondente allo 0 logico, si provveda ad alimentare la scheda di montaggio. Qualcuno degli indicatori luminosi si è acceso?

Noi abbiamo osservato che l'unico indicatore luminoso ad accendersi è stato LM7, in conseguenza del fatto che lo stato normale di $\overline{P0}$ è quello di 1 logico.

Passo 4

Si dispongono gli switch logici SW0, SW1, SW2 ed SW4 secondo diverse combinazioni logiche e si verifichi che gli stati logici corrispondenti siano riprodotti fedelmente dagli indicatori luminosi LM0, LM1, LM2 e LM3.

Passo 5

Si azioni il pulser P0 un certo numero di volte. Cosa osservate?

Noi abbiamo osservato che LM6 e LM7 non si sono mai accesi contemporaneamente, ma sempre o l'uno o l'altro. LM7 era acceso quando l'interruttore con ritorno era nella sua posizione di riposo, o posizione "0", mentre LM6 si accendeva allorchè lo interruttore a ritorno era mantenuto nella sua posizione superiore, ossia "1".

Passo 6

Per quanto riguarda questo esperimento, l'alimentazione della basetta ha qualche importanza?

La risposta è no, poichè con la basetta non sono stati predisposti collegamenti di sorta.

Passo 7

Procederete adesso alla prova delle due file dei bus di alimentazione sulla basetta. Si colleghi con un filo un foro qualsiasi appartenente alla fila del bus esterno con LM5. LM5 si è acceso oppure è rimasto spento?

Noi abbiamo osservato che LM5 si è acceso, ripetere la prova per altri fori della fila esterna. Dovreste osservare tutte le volte l'accensione di LM5.

Passo 8

Per la prova della fila di bus interna si proceda come per il passo 7 collegando con LM5 un qualsiasi foro della fila interna. Si dovrà osservare che LM5 è rimasto spento per qualsiasi foro della fila interna.

Passo 9

Valendovi di un solo indicatore luminoso, come potreste stabilire se un terminale elettrico, un pin di circuito integrato, un reoforo di una resistenza, ecc. si trova al livello logico 0 oppure a quello 1?

Dovreste semplicemente collegare il punto in esame con l'indicatore luminoso. L'accensione dell'indicatore segnala che il punto in esame si trova al livello logico 1, mentre il contrario significa che esso si trova al livello logico 0.

ESPERIMENTO N. 2

Scopo

L'obiettivo di questo esperimento, è dimostrare come si carica il software per esperimenti contenuto in due dispositivi EPROM 2708. Per eliminare l'introduzione da miniterminale degli oltre 2000 bytes che costituiscono il software degli esperimenti di questo libro, la SGS-ATES ha immagazzinato i programmi in due dispositivi EPROM 2708.
Queste EPROM possono essere inserite in due zoccoli della scheda base del Nanocomputer, unitamente alle EPROM del sistema operativo di 2K del Nanocomputer.

Passo 1

Dopo aver tolto tensione al Nanocomputer, inserire i due dispositivi EPROM 2708 negli zoccoli vuoti, adiacenti le EPROM del sistema operativo. Assicurarsi che siano inserite correttamente, operando come segue:

1. Individuare sul dispositivo la tacca di orientamento.
2. Accertarsi che il piedino n. 1 della EPROM contenente il software per gli esperimenti, sia orientato come il piedino n. 1 delle EPROM contenenti il sistema operativo.
3. Assicurarsi che le EPROM siano inserite correttamente, e cioè che non ci siano piedini piegati, che siano tutti inseriti, ecc.

Passo 2

Date tensione al Nanocomputer. Caricate F000 nel registro Program Counter e premete ora il tasto G0.
Cosa osservate?

Noi abbiamo visto comparire attraverso i display a sette segmenti del terminale tastiera/display, il seguente messaggio:

SGS-ATES NANO ROUTINES RELEASE 2- LOADED CIAO

A questo punto la visualizzazione del messaggio termina e viene presentato nel campo indirizzi del display il valore F041 e nel campo dati il valore FF.

Passo 3

Quando il display si spegne, il controllo ritorna al sistema operativo del Nanocomputer. Il programma che genera il messaggio di cui sopra è stato scritto da A. Watts della SGS-ATES, ed inizia nella locazione NANOR2.
Un listing completo e commentato lo trovate nell'appendice. Inoltre, nel Capitolo 5, vi daremo una descrizione completa delle subroutines di ingresso/uscita per la tastiera/display del Nanocomputer, e tutte le informazioni necessarie per poter scrivere voi stessi i programmi di gestione del display. Per ora accontentatevi di questa panoramica.

Passo 4

Esaminate alcune locazioni di memoria a partire dall'indirizzo 0100. Verificate che siano stati caricati i seguenti byte:

Locazione	Contenuto
0100	D3
0101	C5
0102	18
0103	FC
0104	3E
0105	21

Passo 5

Eseguendo il programma dall'indirizzo F000, voi avete iniziato il trasferimento (block move) del contenuto delle EPROM 2708 contenenti le locazioni da F000 a F7FF, nella memoria RAM con inizio nella locazione 0100.
Come potrete osservare negli esperimenti nella parte restante di questo libro, il software è stato caricato e, a meno di alcune modifiche da applicare soltanto in pochi casi, è pronto per l'uso.

Vi diamo ora la mappa della memoria contenuta nelle EPROM.

Salto al programma di trasferimento
Routine per gli esperimenti
Programma di trasferimento
Routine di invio messaggio
Contenuto del messaggio
Bytes delle EPROM non utilizzati

CAPITOLO 3

GENERAZIONE DEGLI IMPULSI DI SINCRONIZZAZIONE: IMPULSI DI SELEZIONE DISPOSITIVI E INDIRIZZI

INTRODUZIONE

Questo capitolo tratta uno dei quattro obiettivi principali dell'interfacciamento dello Z80, la generazione di impulsi di sincronizzazione per l'ingresso e l'uscita di dati, da e verso la CPU. Altri impulsi di sincronizzazione, quali il riconoscimento delle interruzioni e richieste dei bus, verranno trattati nei capitoli successivi. Vi presentiamo un numero adeguato di circuiti di decodifica, compresi alcuni fra quelli che vengono usati dal Nanocomputer per generare i suoi impulsi di selezione dispositivi e di selezione degli indirizzi.

OBIETTIVI

Al termine di questo capitolo, sarete in grado di:

- Dare la definizione di impulsi di selezione dispositvi o di selezione degli indirizzi.
- Dare la definizione dei termini "hardware" e "software".
- Spiegare che cosa si intende per sostituzione dell'hardware con il software.
- Discutere le istruzioni dello Z80 che danno origine alla generazione degli impulsi di selezione dispositivi ed indirizzi.
- Definire ed utilizzare gli impulsi di sincronizzazione della CPU Z80, e cioè $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$ e $\overline{WR}$, ed alcuni segnali quali $\overline{IN}$, $\overline{OUT}$, $\overline{MEMR}$ e $\overline{MEMW}$, derivati dai primi.
- Realizzare schemi di circuiti utilizzabili per generare impulsi di selezione dispositivi ed indirizzi.
- Elencare differenti utilizzazioni degli impulsi di selezione dispositivi.

HARDWARE E SOFTWARE NELL'INTERFACCIAMENTO DELLO Z80

I termini "hardware" e "software" verranno spesso usati in questo capitolo e nei successivi. Diamone quindi una definizione:

Hardware I dispositivi meccanici, magnetici, elettronici ed elettrici di cui è costituito un calcolatore; l'insieme dei materiali costituenti un calcolatore.

Software La totalità dei programmi per ampliare le possibilità operative dei calcolatori, come compilatori, routines e subroutines.

Il calcolatore, insieme ai circuiti integrati, ai fili di collegamento, agli ausilii di cablaggio ed ai dispositivi periferici, vengono tutti considerati hardware. I programmi e le subroutines che usate o scrivete, costituiscono il software. Il sistema operativo del calcolatore è software. Nell'ultimo capitolo, abbiamo definito l'interfacciamento dello Z80 come la sincronizzazione della trasmissione di dati digitali fra la CPU e i dispositivi esterni. Questa sincronizzazione viene quasi sempre attuata attraverso la coordinazione del software con l'hardware. Non è affatto insolito per un programmatore dello Z80 fare spesso riferimento ad uno schema del calcolatore per il quale egli sta creando il software.

L'hardware (rappresentato dallo schema) e il software (rappresentato dal programma) devono interagire al massimo per ottenere i risultati desiderati.

LE ISTRUZIONI DI ACCESSO ALLA MEMORIA E IL SEGNALE $\overline{\text{MREQ}}$

Come abbiamo visto nell'ultimo capitolo, lo Z80 esegue i programmi immagazzinati in memoria leggendo successivamente il codice o i codici operativi dell'istruzione (o fetch), decodificando il codice o i codici operativi, ed eseguendo le operazioni appropriate. Quindi, l'esecuzione di tutte le istruzioni di un programma implica l'interfacciamento con un dispositivo esterno, cioè una memoria. I diagrammi di temporizzazione del ciclo M1 o ciclo di fetch del codice operativo, mostrano come la CPU Z80 comunica al "mondo esterno" che vuole iniziare la lettura dell'istruzione successiva:

1. Viene attivato il segnale $\overline{\text{M1}}$, cioè portato a livello logico 0.
2. L'indirizzo di memoria viene posto sul bus degli indirizzi.
3. Viene attivato il segnale $\overline{\text{MREQ}}$ – questo è temporizzato in modo da andare a 0 logico DOPO che l'indirizzo di memoria si è completamente stabilizzato sul bus degli indirizzi.
4. Viene attivato il segnale $\overline{\text{RD}}$, cioè portato a livello logico 0.

Il "mondo esterno", ed in particolare modo la memoria del sistema, deve ricevere ed interpretare questo messaggio, fare quanto è necessario per ottenere gli otto bit di informazioni, e metterli sul bus dei dati, in modo che quando la CPU legge il bus dei dati stesso, vi trova il dato esatto.

In questo capitolo, ci occuperemo di quello che il "mondo esterno" fa per interpretare i quattro segnali suddetti provenienti dalla CPU. Ma, prima di parlare dell'uso dei suddetti segnali, vediamo come una CPU esegue le istruzioni in un programma. Supponiamo che l'istruzione letta dalla CPU sia LD A,(HL) dove il registro HL contiene 0100. Il codice esadecimale di LD A,(HL) è 7E. Quindi il fetch iniziale del codice operativo, o ciclo M1, eseguito dalla CPU, carica il byte 7E dalla memoria nella CPU. Dopo la decodifica dell'istruzione, la CPU sa che deve eseguire un'istruzione ad un byte (cioè, non sono richiesti altri cicli M1) e che l'istruzione richiede il caricamento nel registro A del contenuto della locazione di memoria puntata dalla coppia di registri HL. Per ottenere questo byte, si richiede un secondo ciclo di lettura in memoria. Per comunicare questa richiesta al "mondo esterno" la CPU pone, sui propri pin di uscita, i seguenti segnali:

1. L'indirizzo della locazione di memoria 0100 viene posto sul bus degli indirizzi
2. Viene attivato il segnale $\overline{\text{MREQ}}$, cioè posto a 0 logico
3. Viene attivato il segnale $\overline{\text{RD}}$, cioè posto a 0 logico.

A parte l'attivazione di $\overline{M1}$, il fetch di un codice operativo fa le stesse cose del ciclo di lettura in memoria iniziato dall'istruzione LD A,(HL). Dato che lo scopo principale del segnale $\overline{M1}$ è quello di associare il byte letto con il codice operativo dell'istruzione in atto le comunicazioni fra la memoria e la CPU vengono di solito coordinate usando i segnali $\overline{MREQ}$, $\overline{RD}$ (per le letture in memoria), $\overline{WR}$ (per le scritture in memoria), il bus dei dati e il bus degli indirizzi . I punti importanti che vorremmo sottolineare sono:

1. Gli accessi in memoria iniziano con l'*esecuzione* delle istruzioni, nonchè con il *fetch* delle stesse
2. I segnali coinvolti nel *fetch* dei codici operativi e nell'*esecuzione* delle istruzioni di lettura in memoria, sono essenzialmente gli stessi.

Supponiamo che la CPU legga l'istruzione LD (HL),A o 77 esadecimale. E' necessario un collegamento con un dispositivo esterno per eseguire questa istruzione? In caso affermativo, quale dispositivo e quali segnali vengono posti sui pin di uscita della CPU?

L'istruzione LD (HL),A richiede che la CPU memorizzi il contenuto dell'accumulatore nella locazione di memoria puntata dal contenuto della coppia di registri HL. Se HL contiene 0100, l'accumulatore deve essere copiato nella locazione 0100, una locazione *esterna* alla CPU. Perciò è chiaramente necessario un collegamento con il "mondo esterno", per cui la CPU pone i seguenti segnali sui propri pin di uscita:

1. 0100 viene posto sul bus degli indirizzi per indicare quale locazione di memoria deve essere scritta
2. Viene attivato il segnale $\overline{MREQ}$, dato che il dispositivo esterno è una memoria
3. Il contenuto dell'accumulatore viene posto sul bus dei dati per essere copiato nella locazione di memoria 0100

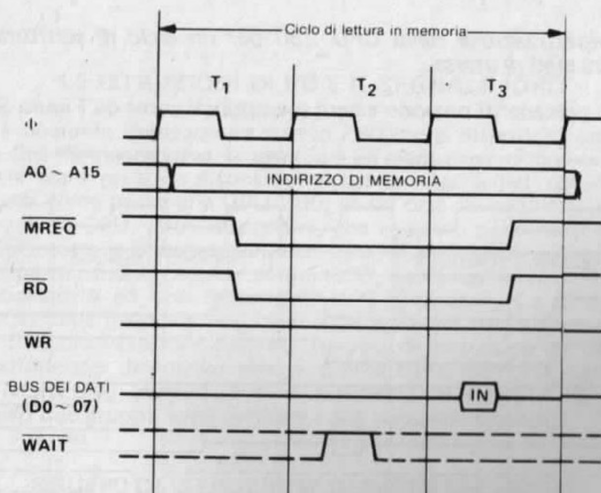


Figura 3-1. Temporizzazione della CPU Z80 per un ciclo di lettura in memoria senza stati di attesa.

4. Viene attivato il segnale $\overline{WR}$, dato che l'operazione desiderata è un'operazione di scrittura (WRite).

Come potete vedere esistono delle differenze fra la lettura da memoria e la scrittura in memoria. Innanzitutto, la lettura causa l'attivazione del segnale $\overline{RD}$, mentre la scrittura provoca l'attivazione del segnale $\overline{WR}$. Un'altra differenza più sottile è costituita dalla temporizzazione per il posizionamento dei dati sul bus dei dati. In un'operazione di lettura, il dispositivo esterno pone i dati sul bus dei dati affinchè la CPU li possa leggere. In un'operazione di scrittura, è la CPU a porre i dati sul bus dei dati affinchè il dispositivo esterno li legga. I diagrammi di temporizzazione di queste due operazioni, che potete vedere nelle Figure 3-1 e 3-2, mostrano molto chiaramente le differenze di temporizzazione.

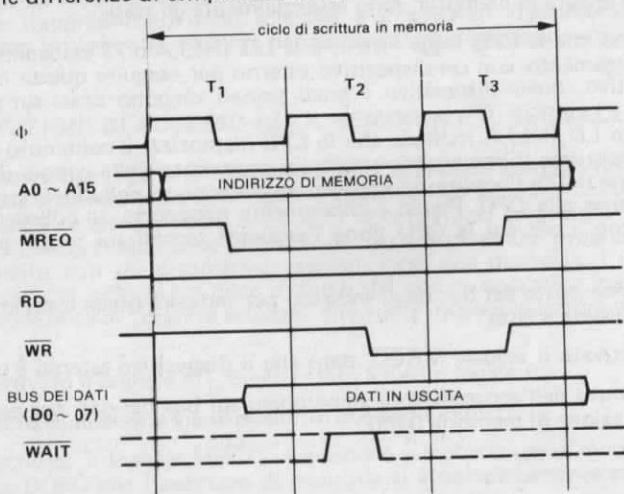


Figura 3-2. Temporizzazione della CPU Z80 per un ciclo di scrittura in memoria senza stati di attesa.

I due diagrammi precedenti possono essere sovrapposti come da Figura 3-3.

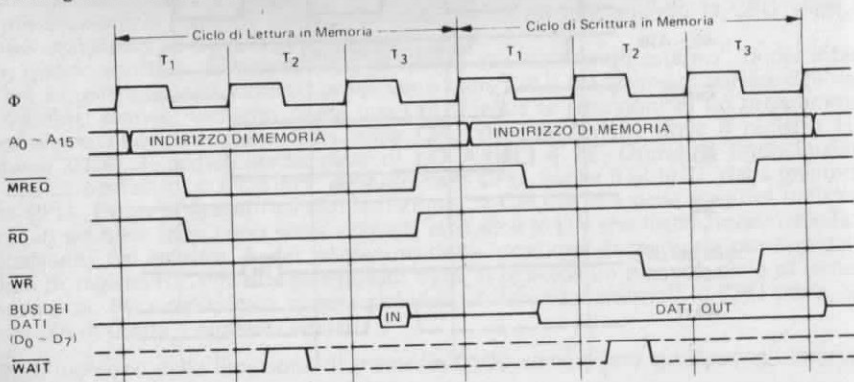


Figura 3-3. Temporizzazione della CPU Z80 per un ciclo di lettura e scrittura in memoria senza stati di attesa.

La sovrapposizione è una tecnica molto comune, utilizzata per facilitare il confronto delle temporizzazioni. Precedentemente, abbiamo fatto menzione delle sottili differenze di temporizzazione tra le operazioni di lettura e scrittura in memoria. In particolare, la temporizzazione relativa al posizionamento dei dati sul bus dei dati è l'attivazione degli impulsi di sincronizzazione $\overline{MREQ}$ e $\overline{WR}$ o $\overline{RD}$, è critica. Si nota che per una operazione di lettura, $\overline{MREQ}$ e $\overline{RD}$ sono attivi ben prima che la memoria ponga i dati sul bus dei dati. Al contrario, $\overline{WR}$ è attivo solo 100 nanosecondi dopo il posizionamento dei dati sul bus dei dati durante una operazione di scrittura.

In una operazione di scrittura, è critico che il fatto che dati ed indirizzi corretti siano sui bus dati ed indirizzi *prima* che si dia alla memoria il comando di SCRITTURA. Dato che gli indirizzi ed i dati sono normalmente bufferizzati, e gli indirizzi sono decodificati, si verificano degli inevitabili ritardi. Quindi, per una corretta operatività dei chip di memoria, occorre ritardare il segnale $\overline{WR}$. Per assicurarsi che arrivi al pin "R/W" della memoria dinamica, *dopo* i dati e gli indirizzi validi. Un segnale, chiamato $\overline{DBWR}$ (delayed-buffered-write-bar ossia abilitazione scrittura bufferato e ritardato) è generato sul Nanocomputer e trasferito sulla piastra di cablaggio del Nanocomputer, proprio per far fronte a questi problemi.

I segnali $\overline{DBWR}$ e $\overline{BWR}$ sono ritardati di circa 100 nanosecondi. Si ricordi che $\overline{BWR}$ è la versione bufferizzata di $\overline{WR}$. E' in genere più sicuro usare $\overline{DBWR}$ in tutti gli esperimenti di questo testo.

Sentitevi comunque senz'altro liberi di passare da $\overline{DBWR}$ a $\overline{BWR}$ per verificare eventuali differenze di prestazioni. Il circuito per generare $\overline{DBWR}$ da $\overline{BWR}$ è dato in Figura 2-8.

Verifichiamo ora se avete capito quanto è stato detto fin qui: che cosa succede se la CPU legge l'istruzione LD A,B?

A parte il ciclo di fetch del codice operativo della memoria, è necessario che lo Z80 venga interfacciato con un dispositivo esterno per eseguire questa istruzione? Sui pin di uscita dello Z80, vengono posti dei segnali di sincronizzazione durante l'esecuzione di questa istruzione? La riposta ad entrambe le domande è no, perchè A e B si riferiscono a registri *interni* alla CPU, perciò la copiatura del contenuto del registro B nel registro A è un'operazione per cui la CPU non richiede nessun aiuto esterno.

LE ISTRUZIONI DI I/O E IL SEGNALE $\overline{IORQ}$

Volendo, è possibile interfacciare con lo Z80 molti dispositivi esterni, diversi da una memoria. Sul Nanocomputer, la tastiera è un dispositivo d'ingresso, ed i display a LED ed a sette segmenti sono dispositivi di uscita interfacciati con la CPU Z80. Se in una lavapiatti viene posta una CPU Z80, in tal caso la lavapiatti è un dispositivo interfacciato con la CPU. Altri dispositivi, che vengono comunemente interfacciati con le CPU Z80, sono le telescriventi, altri Z80, display grafici, lettori e perforatori di banda, stampanti, dischi, cassette magnetiche, apparecchiature musicali, strumentazione di laboratorio ed una miriade di altri dispositivi. La CPU Z80 distingue questi dispositivi dalla memoria, associando un software particolare alla generazione degli impulsi di sincronizzazione richiesti. I gruppi di istruzione che generano impulsi di sincronizzazione per dispositivi esterni diversi dalla memoria, vengono chiamati GRUPPO d'INGRESSO (Figura 3-4) e GRUPPO di USCITA (Figura 3-6). Ad ognuno di questi due gruppi, sono dedicati i due paragrafi seguenti.

GRUPPO DI ISTRUZIONE DI INGRESSO (INPUT)

Come potete vedere, lo Z80, al contrario dell'8080, la cui CPU implementa solo un'istruzione d'ingresso (Codice operativo DB), ha capacità di ingresso estese. Esami-

DESTINAZIONE			INDIRIZZO DELLA PORTA SORGENTE: IMMEDIATO (n)	INDIRIZZO DELLA PORTA SORGENTE: INDIRETTO (C)	
INPUT 'IN'	REGISTRO	A	DB n	ED 78	
		B		ED 40	
		C		ED 48	
		D		ED 50	
		E		ED 58	
		H		ED 60	
		L		ED 68	
'INI' – INPUT & Inc HL, Dec B	INDIRETTO	(HL)		ED A2	COMANDI DI INPUT DI BLOCCHI
'INIR' – INP, Inc HL, Dec B, RIPETE SE B ≠ 0	INDIRETTO	(HL)		ED B2	COMANDI DI INPUT DI BLOCCHI
'IND' – INPUT & Dec HL, Dec B	INDIRETTO	(HL)		ED AA	COMANDI DI INPUT DI BLOCCHI
'INDR' – INPUT, Dec HL, Dec B, RIPETE SE B ≠ 0	INDIRETTO	(HL)		ED BA	COMANDI DI INPUT DI BLOCCHI

Figura 3–4. Gruppo di istruzioni di ingresso.

niamo ora da vicino ciascuna delle istruzioni d'ingresso dello Z80. Tenete presente che useremo le seguenti abbreviazioni:

r è uno qualunque dei registri A, B, C, D, E, H o L
I seguenti codici a tre bit vengono incorporati nel codice operativo, per indicare:

r	Registro
000	B
001	C
010	D
011	E
100	H
101	L
110	non usato (Set di Flag)
111	A

n è un numero esadecimale a due digit (8 bit).

IN A,(n)

Questa istruzione carica nell'accumulatore otto bit di dati provenienti dal dispositivo esterno, il cui codice dispositivo è stato indicato con n. Il codice esadecimale di questa istruzione è DB n, dove n è il codice dispositivo esadecimale a due digit. Eseguendo questa istruzione, la CPU Z80 effettua le seguenti operazioni:

1. Pone il codice dispositivo n sugli otto bit inferiori del bus degli indirizzi (A0-A7). Pone il contenuto dell'accumulatore sugli otto bit superiori del bus degli indirizzi (A8-A15).
2. Attiva i segnali $\overline{IORQ}$ e $\overline{RD}$.
3. Legge nell'accumulatore i dati sul bus dei dati, qui posizionati dal dispositivo n, in risposta ai segnali suddetti.

La temporizzazione per l'esecuzione di queste operazioni è data dal diagramma di temporizzazione del ciclo d'ingresso dello Z80, in Figura 3-5.

L'istruzione IN A,(n) è la sola istruzione d'ingresso implementata dal microprocessore 8080A della Intel e anzichè porre il contenuto dell'accumulatore sulla metà superiore del bus degli indirizzi, l'8080A duplica il codice dispositivo su A8-A15. Questa differenza si evidenzia negli aspetti hardware dell'interfacciamento dell'8080A rispetto allo Z80, ma in genere, non crea alcun problema riguardo alla compatibilità software tra le due CPU. Non ci è mai capitato di vedere schemi riguardanti i circuiti d'ingresso sia dell'8080A che dello Z80 in cui questa differenza causasse delle incompatibilità a livello software.

Come vedrete, tutte le istruzioni di ingresso e di uscita dello Z80 fanno uso degli otto bit superiori del bus degli indirizzi invece di duplicare semplicemente gli otto bit inferiori. Questo viene fatto con l'obiettivo di mettere a disposizione, della circuiteria esterna che fa da supporto alla CPU, il maggior numero di informazioni possibile. In questo caso, lo stato dell'accumulatore prima della esecuzione dell'istruzione IN A,(n) viene posto in uscita sulla metà superiore del bus degli indirizzi, come porzione di informazione che può essere usata dai circuiti di interfaccia. Queste informazioni possono essere usate, ad esempio, per:

- Scambiare i byte con il dispositivo d'ingresso, se il dispositivo stesso può svolgere un'operazione di questo tipo.
- Fornire impulsi di strobe ad altri circuiti, per eseguire operazioni contemporanee all'operazione d'ingresso, quali reset dei flag di stato, sospendere le operazioni in corso, iniziare operazioni collegate, fornire impulsi di strobe a contatori o timer, ecc.

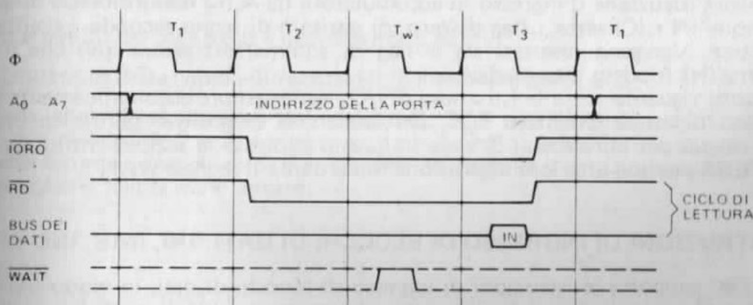


Figura 3-5. Temporizzazione di un ciclo di ingresso. Lo stato di attesa T_W è inserito automaticamente.*

IN r,(C)

Questa istruzione aggiunge una nuova dimensione di flessibilità alle operazioni di ingresso dati, ovvero la capacità di usare l'indirizzamento indiretto tramite un registro, per indicare i codici dispositivo. Ciò significa che è possibile indirizzare il dispositivo d'ingresso desiderato caricando il registro C con il suo codice dispositivo Un'altra caratteristica per cui questa istruzione si rivela più flessibile della precedente, è la possibilità di specificare il registro destinazione dei dati in ingresso. Per esempio:

ED 40 IN B,(C)

carica nel registro B i dati provenienti dal dispositivo indirizzato dal codice a otto bit contenuto nel registro C. Le operazioni specifiche svolte dalla CPU Z80 durante l'esecuzione dell'istruzione IN r, (C) sono:

1. Porre il contenuto del registro C sugli otto bit inferiori del bus degli indirizzi (A0-A7).
   Porre il contenuto del registro B sugli otto bit superiori del bus degli indirizzi (A8-A15).
2. Attivare i segnali $\overline{IORQ}$ e $\overline{RD}$.
3. Leggere nel registro r i dati posti sul bus dei dati dal dispositivo esterno indirizzato dal codice dispositivo presente nel registro C.

La temporizzazione per l'esecuzione di queste operazioni è data dal diagramma di temporizzazione che illustra il ciclo d'ingresso dello Z80, in Figura 3-5. Il codice operativo dell'istruzione IN r,(C) è di due byte.

Il primo byte è sempre ED, mentre il secondo byte è determinato dal caricamento nei bit D3, D4, D5 del codice del registro corrispondente al registro destinazione r. Quindi, il secondo byte è:



Se si usa il codice a tre bit 110, che non ha alcuna corrispondenza con un registro a otto bit, l'istruzione risultante, ED 70, cambia soltanto il registro dei flag e non influenza nessun altro registro.

Mentre l'istruzione d'ingresso in accumulatore IN A,(n) non influenza nessun flag, l'istruzione IN r,(C) setta i flag di zero, di parità e di segno secondo i risultati della operazione. Vengono resettati sia il flag di somma/sottrazione (N) che il flag di half-carry (H) (riporto intermedio).
Per quanto riguarda tutte le istruzioni di I/O, viene sempre automaticamente inserito uno stato di attesa chiamato T_W*, per lasciare al dispositivo periferico un po' di tempo in più per la risposta. Si veda in questo capitolo la sezione intitolata STATI di ATTESA per una ulteriore discussione riguardante il segnale WAIT.

LE ISTRUZIONI DI INGRESSO DI BLOCCHI DI DATI: INI, INIR, IND e INDR

Lo Z80 supporta le istruzioni di ingresso di blocchi di dati, in modo analogo alle istruzioni di spostamento e di ricerca blocchi, trattate nel Volume 1. Prima di eseguire una qualunque di queste istruzioni, un programma dello Z80 deve *"inizializzare"* i registri B, C e HL nel modo che segue.

B = numero dei byte da inserire
C = codice del dispositivo d'ingresso
HL = indirizzo d'inizio per la stringa dei byte in ingresso.

INI

Eseguendo l'istruzione INI *("ingresso-incremento")*, avviene quanto segue:

1. Un byte proveniente dal dispositivo indirizzato dal registro C viene caricato nella locazione di memoria puntata dal contenuto della coppia di registri HL. A livello della CPU e dei circuiti di interfaccia:
   a. Il codice dispositivo presente nel registro C è posto sulla metà inferiore del bus degli indirizzi (A0-A7).
   Il contenuto del registro B, cioè il contatore dei byte, è posto sulla metà superiore del bus degli indirizzi (A8-A15).
   b. Vengono attivati i segnali $\overline{IORQ}$ e $\overline{RD}$.
   c. La CPU legge il byte di dati posto sul bus dei dati dal dispositivo periferico, in risposta ai suddetti segnali. Il byte in ingresso è memorizzato nella locazione di memoria puntata da HL. (Notate che la memorizzazione del byte in ingresso in (HL) dà inizio ad un ciclo di scrittura in memoria!)
2. Il contenuto del registro B è decrementato di uno.
3. Il contenuto della coppia di registri HL è incrementato di uno.
4. Se il registro B contiene 0 dopo essere stato decrementato nel Passo 2, viene settato il flag di zero, altrimenti, viene resettato. Viene settato il flag di somma/sottrazione (N).

INIR

Con l'esecuzione dell'istruzione INIR *("ingresso-incremento-ripetizione")* si verifica quanto segue:

1. Un byte proveniente dal dispositivo indirizzato dal contenuto del registro C viene caricato nella locazione di memoria puntata dal contenuto della coppia di registri HL. In breve,

   (HL) ⟵ (C)

   La CPU ed i circuiti di interfaccia operano nello stesso modo descritto da a fino a c, come sopra.
2. Il contenuto del registro B è decrementato di uno.
3. Il contenuto della coppia di registri HL è incrementato di uno.
4. Viene controllato il valore del registro B. Se B non è uguale a zero, vengono ripetuti i Passi 1, 2 e 3, e viene resettato il flag di zero. Se B è uguale a zero viene settato il flag di zero, e si procede ad eseguire l'istruzione successiva. In ogni caso il flag N viene settato.

IND e INDR

Le esecuzioni delle istruzioni IND *("ingresso-decremento")* e INDR *("ingresso-decremento-ripetizione")* risultano molto simili fra di loro. L'unica differenza sta nel fatto che il Passo 3 *decrementa* HL. Quindi INI e INIR caricano le stringhe

di byte in memoria verso l'alto, mentre le istruzioni IND e INDR caricano le stringhe di byte verso il basso della memoria.

Invece di farvi molti esempi sull'uso delle istruzioni del GRUPPO D'INGRESSO, supporremo che ne abbiate pienamente afferrato l'utilità e passeremo a discutere del GRUPPO DI USCITA (OUTPUT). In questo capitolo, la nostra intenzione è di mettere in evidenza gli aspetti dell'interfacciamento che riguardano l'hardware, e in particolare i circuiti necessari per decodificare i segnali che la CPU pone in uscita come risultato dell'esecuzione delle istruzioni IN e OUT.

Questa è la ragione per cui la descrizione che faremo del GRUPPO DI USCITA è analoga a quella fatta per il GRUPPO D'INGRESSO. Rimanderemo gli esempi relativi al software ai capitoli successivi e rivolgeremo l'attenzione ai segnali generati sul bus dei dati, su quello degli indirizzi e su quello di controllo.

GRUPPO DI ISTRUZIONI DI USCITA

Come potete vedere, il GRUPPO DI USCITA comprende le istruzioni che sono il corrispondente logico delle istruzioni d'ingresso per le uscite. Qui di seguito vi elenchiamo le istruzioni di uscita, con le istruzioni d'ingresso corrispondenti:

			SORGENTE								
			REGISTRO							INDIRETTO	
			A	B	C	D	E	H	L	(HL)	
'OUT'	IMMEDIATO	(n)	D3 n								
	INDIRETTO	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69		
'OUTI' – OUTPUT Inc HL, Dec b	INDIRETTO	(C)								ED A3	COMANDI DI OUTPUT DI BLOCCHI
'OTIR' – OUTPUT, Inc HL, Dec B, RIPETE SE B ≠ 0	INDIRETTO	(C)								ED B3	
'OUTD' – OUTPUT Dec HL & B	INDIRETTO	(C)								ED AB	
'OTDR' – OUTPUT, Dec HL & B, RIPETE SE B ≠ 0	INDIRETTO	(C)								ED BB	

INDIRIZZO DELLA PORTA DESTINAZIONE

Figura 3-6. Gruppo di istruzioni di uscita.

Istruzioni di uscita	Istruzioni d'ingresso corrispondenti
OUT (n),A	IN A,(n)
OUT (C),r	IN R,(C)
OUTI	INI
OTIR	INIR
OUTD	IND
OTDR	INDR

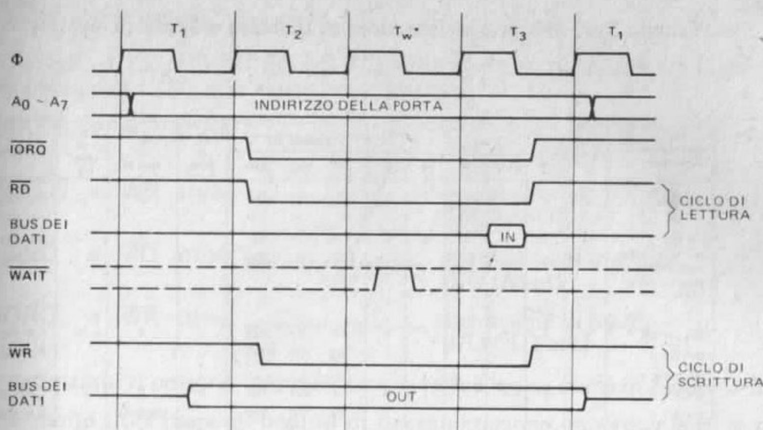


Figura 3-7. Temporizzazione dei cicli di ingresso e di uscita della CPU Z80.

Vi sono tre differenze fondamentali fra il gruppo di istruzioni d'ingresso e il gruppo di istruzioni di uscita:

1. La direzione del flusso dei dati rispetto alla CPU.
2. Un'istruzione di uscita attiva il segnale $\overline{WR}$, laddove la istruzione di ingresso corrispondente attiva il segnale $\overline{RD}$.
3. Esistono differenze nelle temporizzazioni dovute al diverso ruolo tenuto dalla CPU come ricevitore (ingresso) o trasmettitore (uscita). La Figura 3-7 mostra la temporizzazione per i cicli di ingresso e di uscita sullo stesso diagramma. Come potete vedere, la CPU pone i dati in uscita sul bus dei dati per un tempo relativamente lungo, se confrontato con quello in cui un dispositivo periferico pone i suoi dati sullo stesso bus. Perciò, durante un'operazione d'ingresso, la CPU ha un margine di errore molto minore rispetto ad un'unità periferica impegnata a prelevare i dati dal bus dei dati, nel corso di un'operazione di uscita dei dati stessi dalla CPU. La temporizzazione relativa al segnale $\overline{IORQ}$ è invece la stessa per entrambi.
4. A causa dei possibili requisiti in termini di temporizzazioni relative nei microprocessori reali, in rapporto ai tempi di arrivo dei dati e degli impulsi di sincronizzazione durante i cicli di scrittura di I/O, può rendersi necessario un segnale $\overline{WR}$ ritardato ($\overline{DBWR}$ nella scheda di cablaggio del Nanocomputer). La discussione relativa alla desiderabilità del segnale $\overline{DBWR}$ per i cicli di scrittura in memoria, si applica parimenti ai cicli di scrittura di I/O.

In Figura 3-7 i segnali per l'ingresso e l'uscita sono sovrapposti per comodità, dato che i cicli macchina di lettura e scrittura non possono realizzarsi come mostrato, uno dentro l'altro. I cicli di attesa T_W*, automaticamente inseriti, sono descritti nella sezione di questo capitolo intitolata STATI DI ATTESA (WAIT STATES).

In Tabella 3-1 è data una descrizione di rapida consultazione, per ogni istruzione di ingresso ed uscita; la tabella è tratta dallo Z80 Technical Manual della SGS-ATES. Ricordando le funzioni delle analoghe istruzioni di ingresso, dovreste essere in grado di comprendere completamente ciascuna istruzione di uscita.

Tabella 3-1. Gruppo di istruzioni di ingresso e di uscita.

Codice mnemonico	Operazione Simbolica	Flag C	Flag Z	Flag P/V	Flag S	Flag N	Flag H	Codice OP 76	Codice OP 543	Codice OP 210	N. di Byte	N. di cicli M	N. di stati T	Commenti
IN A, (n)	A ← (n)	•	•	•	•	•	•	11 ←	011 n	011 →	2	3	11	n su $A_0 \sim A_7$ Acc su $A_8 \sim A_{15}$
IN r, (C)	r ← (C) se r = 110 solamente i flag saranno modificati	•	↕	P	↕	0	↕	11 01	101 r	101 000	2	3	12	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	•	① ↕	X	X	1	X	11 10	101 100	101 010	2	4	16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Ripete finché B = 0	•	1	X	X	1	X	11 10	101 110	101 010	2 2	5 (Se B≠0) 4 (Se B= 0)	21 16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL-1	•	① ↕	X	X	1	X	11 10	101 101	101 010	2	4	16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL-1 Ripete finché B = 0	•	1	X	X	1	X	11 10	101 111	101 010	2 2	5 (Se B≠0) 4 (Se B= 0)	21 16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	•	•	•	•	11 ←	010 n	011 →	2	3	11	n su $A_0 \sim A_7$ Acc su $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	•	•	•	•	11 01	101 r	101 001	2	3	12	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B - 1 HL ← HL+1	•	① ↕	X	X	1	X	11 10	101 100	101 011	2	4	16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B - 1 HL ← HL+1 Ripete finché B = 0	•	1	X	X	1	X	11 10	101 110	101 011	2 2	5 (Se B≠0) 4 (Se B=0)	21 16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL-1	•	① ↕	X	X	1	X	11 10	101 101	101 011	2	4	16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B-1 HL ← HL-1 Ripete finché B = 0	•	1	X	X	1	X	11 10	101 111	101 011	2 2	5 (Se B≠ 0) 4 (Se B= 0)	21 16	C su $A_0 \sim A_7$ B su $A_8 \sim A_{15}$

Note: ① Se il risultato di B-1 è zero il flag Z viene posto ad uno, altrimenti è posto a zero.

Flag: • = flag non modificato 0 = reset del flag 1 = set del flag X = flag posizionato casualmente
↕ = il flag viene modificato in accordo col risultato dell'operazione.

I Segnali $\overline{IN}$, $\overline{OUT}$, $\overline{MEMR}$, $\overline{MEMW}$

Il significato di $\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$ e $\overline{IORQ}$, nella sincronizzazione dell'I/O controllato da programma dello Z80, può essere così riassunto:

- $\overline{MREQ}$ e $\overline{RD}$ attivo ———————— operazione di lettura in memoria (MEMory Read)
- $\overline{MREQ}$ e $\overline{WR}$ attivo ———————— operazione di scrittura in memoria (MEMory Write)
- $\overline{IORQ}$ e $\overline{RD}$ attivo ———————— operazione di ingresso (I/O INput)
- $\overline{IORQ}$ e $\overline{WR}$ attivo ———————— operazione di uscita (I/O OUTput)

Queste definizioni si possono rappresentare in forma logica come in Figura 3-8.

Spesso è molto utile usare gli impulsi di sincronizzazione, rappresentanti le quattro precedenti situazioni, in tutti i tipi di circuiti di I/O per microcomputer.
Una tabella della verità come la Tabella 3-2, per il segnale $\overline{MEMR}$, mostra chiaramente il rapporto tra le due definizioni per ciascun segnale dato precedentemente.
Ad esempio, la tabella mostra che $\overline{MEMR}$ è attivo (0 logico) solo quando sia $\overline{MREQ}$ che $\overline{RD}$ sono attivi (0 logico).

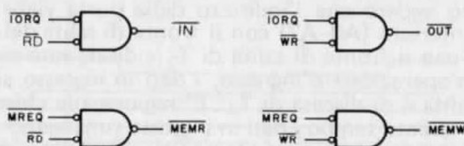


Figura 3-8. I segnali $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IN}$ e $\overline{OUT}$.

Tabella 3-2. Tabella della verità per la generazione del segnale $\overline{MEMR}$.

MREQ	RD	MEMR
0	0	0 (attivo)
0	1	1
1	0	1
1	1	1

Per riassumere alcuni dei punti essenziali delle ultime due sezioni:

- Lo Z80 legge la memoria esterna e poi comunica con un altro dispositivo, nel ciclo di fetch del codice operativo, o M1, per eseguire ogni istruzione del programma in memoria.
- Vi sono due classi principali di istruzioni dello Z80, la cui esecuzione richiede alla CPU di interfacciarsi con dispositivi esterni:
  a. La classe $\overline{MREQ}$: istruzioni che accedono alla memoria per leggere ($\overline{RD}$) o scrivere ($\overline{WR}$) come LD (HL),A INC (HL) o AND (HL)
  b. La classe $\overline{IORQ}$: i gruppi di istruzioni $\overline{IN}$ e $\overline{OUT}$.

- Tutte le istruzioni che richiedono alla CPU di interfacciarsi con un dispositivo esterno pongono o un indirizzo o un codice dispositivo sul bus degli indirizzi; attivano due dei segnali di controllo $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$ e leggono o scrivono dati sul bus dei dati.
- Le operazioni di scrittura e di uscita possono richiedere l'uso di un segnale $\overline{WR}$ ritardato.
- Quattro segnali intermedi molto utili, con riguardo all'accesso in memoria od ai circuiti di I/O, sono:

$$\overline{MEMR} = \overline{MREQ} \cdot \overline{RD}$$
$$\overline{MEMW} = \overline{MREQ} \cdot \overline{WR}$$
$$\overline{IN} = \overline{IORQ} \cdot \overline{RD}$$
$$\overline{OUT} = \overline{IORQ} \cdot \overline{WR}$$

IL CALCOLO DELLE TEMPORIZZAZIONI PER L'I/O DELLO Z80

In questo capitolo ed in quelli precedenti, avete visto molti diagrammi di temporizzazione. Questi diagrammi mostrano soltanto i tempi *relativi* di realizzazione di un dato insieme di eventi, cioè i segnali sui tre bus esterni dello Z80. Ad esempio, nel diagramma di temporizzazione delle operazioni di ingresso e di uscita della Figura 3-7, possiamo vedere che l'indirizzo della porta viene posto sul bus degli indirizzi nella metà inferiore (A0-A7) con il fronte di salita del ciclo-TT_1, il segnale $\overline{IORQ}$ viene attivato con il fronte di salita di T_2 e disattivato con il fronte di discesa di T_3; e, durante un'operazione d'ingresso, i dati in ingresso si trovano sul bus dei dati fra i fronti di salita e di discesa di T_3. E' ragionevole chiedersi quanto duri un impulso $\overline{IORQ}$, o per quanto tempo i dati in ingresso rimangono sul bus dati.

Per rispondere a questa e ad altre domande sul tempo *assoluto*, è necessaria solo un'informazione, cioè la lunghezza di un ciclo-T o delle sue frazioni. Il tempo di ogni ciclo-T è l'inverso della frequenza di clock del Nanocomputer. Per esempio, se il vostro calcolatore opera alla velocità di 1 MHz o di 1.000.000 cicli di clock al secondo, il tempo del ciclo è di 1/1.000.000 di secondi per ogni ciclo-T, ossia 1 microsecondo. Lo Z80 più veloce opera a 4 MHz o 250 nanosecondi (0,250 microsecondi) per ogni ciclo-T. Il Nanocomputer opera a circa 2,5 MHz, quindi quanto è lungo un ciclo-T? (Risposta: 400 nanosecondi). Per rispondere ad altri interrogativi suscitati dalla precedente trattazione, per un Nanocomputer che opera a 2,5 MHz, l'impulso $\overline{IORQ}$ dura 2,5 cicli-T, o 1000 nanosecondi; i dati in ingresso restano sul bus dei dati per poco meno della metà di un ciclo-T, o 200 nanosecondi. Come si vede non si tratta di tempi lunghi.

Quando caricate i vari digit dalla tastiera, per ottenere che il Nanocomputer "vi ascolti" è necessario letteralmente spaccare il secondo.

Questa precisione nella sincronizzazione dei segnali è alla base di tutti i calcolatori digitali. Man mano che gli sviluppi tecnologici ci permettono di suddividere i secondi in frazioni sempre più piccole, le temporizzazioni possono diventare sempre più strette.

La nostra tecnologia è arrivata al punto in cui la lunghezza dei collegamenti che conducono i segnali da un componente di un calcolatore, come la memoria esterna, ad un'altra area, come i registri di una CPU, è diventata un fattore significativo, perchè una lunghezza pari a un piede, significa un ritardo di trasmissione di un nanosecondo. Speriamo che, leggendo questo paragrafo, vi siate resi conto della sorprendente velocità alla quale opera il Nanocomputer, e della necessità di una precisa sincronizzazione dei fenomeni che si verificano sia all'interno che all'esterno della CPU.

GLI STATI DI ATTESA (WAIT STATES)

Un modo per ovviare alle restrizioni delle temporizzazioni, di cui abbiamo precedentemente parlato, consiste nell'uso degli *stati di attesa.* Gli stati di attesa sono praticamente dei cicli-T *"non corrispondenti ad eventi"* ma inseriti in un ciclo macchina per concedere tempi più lunghi allo svolgersi di operazioni o procedimenti critici, durante questi cicli lo stato di tutti i segnali è mantenuto costante.
Per esempio, durante un ciclo di lettura in memoria senza stati di attesa (si veda la Figura 3-1), il tempo che intercorre da quando la CPU fornisce impulsi di strobe ad una locazione di memoria a quando la CPU stessa legge il bus dei dati è circa due cicli-T, o 500 nanosecondi con un clock a 4 MHz. Il *TEMPO DI ACCESSO* alla memoria potrebbe non essere così veloce. Il *tempo di accesso* è il tempo richiesto dalla memoria per "sentire" la sua selezione, relativa ad una comunicazione con la CPU, per trovare il byte di dati indirizzato e porlo sul bus dei dati.

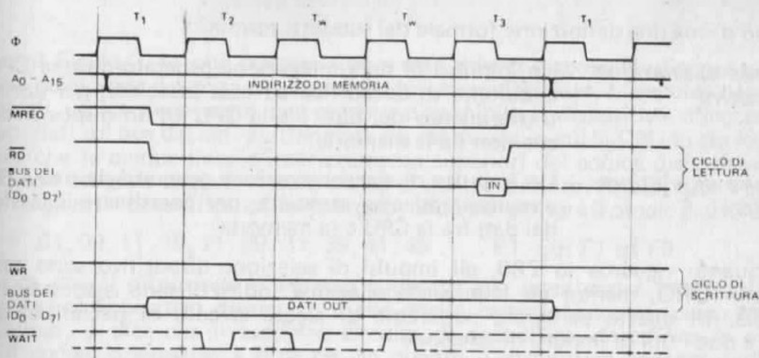


Figura 3-9. Cicli di lettura o scrittura in memoria, con stati di attesa, della CPU Z80.

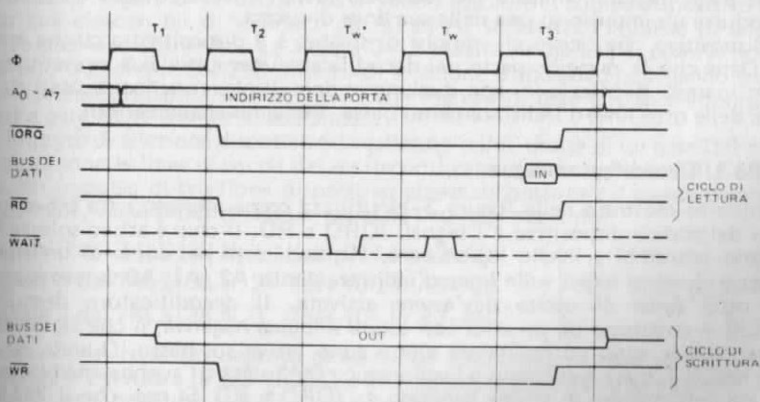


Figura 3-10. Cicli di ingresso o uscita, con stati di attesa, della CPU Z80.

Supponiamo che la memoria possa porre i dati presenti in uno dei suoi indirizzi sul bus dei dati in 1,8 microsecondi. Usando il pin $\overline{WAIT}$ dello Z80, la memoria può dire alla CPU di aspettare per altri due cicli-T prima di leggere il bus dei dati. La circuiteria che interfaccia la memoria con la CPU contiene la logica per mantenere il segnale $\overline{WAIT}$ attivo per la durata dei due cicli-T; la CPU, scoprendo che la sua linea di attesa è bassa "ozia" finchè la suddetta linea non si alza nuovamente, leggendo quindi i dati. La CPU controlla dapprima la sua linea $\overline{WAIT}$ al fronte di discesa di T_2 del ciclo di lettura in memoria, e quindi effettua un controllo per ogni ciclo-T successivo. Vi mostriamo nelle Figure 3-9 e 3-10 i diagrammi di temporizzazione, in cui sono inseriti gli stati di attesa, relativi ai cicli di accesso alla memoria e ai dispositivi di I/O.

COME SI GENERANO GLI IMPULSI DI SELEZIONE DISPOSITIVO E INDIRIZZI

Diamo prima una definizione formale dei suddetti termini:

Impulso di selezione dispositivo — Un impulso di sincronizzazione generato da una CPU e dalla circuiteria di decodifica ad essa associata, per coordinare il trasferimento dei dati fra la CPU ed un dispositivo esterno, che non sia la memoria.

Impulso di selezione indirizzi — Un impulso di sincronizzazione generato da una CPU e dalla circuiteria ad essa associata, per coordinare il trasferimento dei dati fra la CPU e la memoria.

Per quanto riguarda lo Z80, gli impulsi di selezione dispositivo sono associati al segnale $\overline{IORQ}$, mentre gli impulsi di selezione indirizzi sono associati al segnale $\overline{MREQ}$. In questo paragrafo parleremo di alcuni circuiti di decodifica relativi a tutti e due i tipi di impulsi di selezione.

Impulsi di selezione dispositivo

La circuiteria per la generazione degli impulsi di selezione dispositivo dello Z80 solitamente utilizza un decodificatore/demultiplexer e alcune porte diverse che combinano fra loro i segnali $\overline{IORQ}$ e $\overline{RD}$ o $\overline{WR}$ per lo strobe degli otto bit inferiori del bus degli indirizzi, impulsi che vengono inviati al decodificatore/demultiplexer per produrre un impulso su una delle sue linee di uscita.

Il dispositivo che riceve gli impulsi di strobe, è il dispositivo a cui ha accesso la CPU. Dato che la maggior parte dei decodificatori/demultiplexer presentano meno di otto ingressi di selezione, con qualunque demultiplexer vengono usati dei sottoinsiemi delle otto linee d'indirizzamento basse. Vediamo alcuni esempi.

Esempio 1: Decodifica ambigua

Il circuito mostrato nella Figura 3-11 utilizza come ingresso i tre bit meno significativi del codice dispositivo e i segnali $\overline{IORQ}$ e $\overline{RD}$. Il chip è attivo solo se $\overline{IORQ}$ e $\overline{RD}$ sono entrambi a livello logico zero, e quindi solo nel corso di un'operazione d'ingresso. I valori logici sulle linee d'indirizzamento A2, A1, A0 determinano quale delle otto linee di uscita dev'essere attivata. Il decodificatore demultiplexer 74LS138 è costituito da un chip con uscita a logica negativa, il che significa che le sue otto uscite sono normalmente alte e sono attive sul basso. Quindi, se A0 è a livello logico 1 e A1 e A2 sono a livello logico 0, l'uscita 01 si abbasserebbe per tutta la durata dell'impulso di strobe generato da $\overline{IORQ}$ e $\overline{RD}$. Si noti che il 74LS138 ha tre ingressi di abilitazione in AND; di questo due sono attivi sul basso e possono essere direttamente collegati con $\overline{IORQ}$ e $\overline{RD}$.

Riportiamo nella Tavola 3-1 una descrizione del decodificatore/demultiplexer 74LS138 da 3 a 8 linee, come compare nel data Book della SGS-ATES.

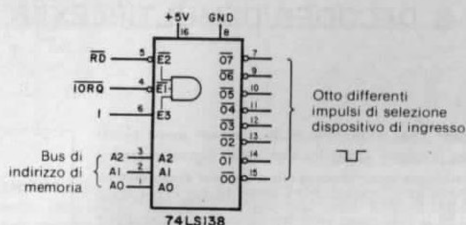


Figura 3-11. Come generare otto differenti impulsi di selezione dispositivo di ingresso con un Decodificatore/Demultiplexer 74LS138.

Il circuito di Figura 3-11 può trattare fino a otto diversi dispositivi d'ingresso, uno per ognuna delle linee di uscita del decodificatore/demultiplexer. L'impulso negativo sulla linea di uscita viene usato per rendere disponibile il dispositivo d'ingresso a porre i suoi dati sul bus dei dati esattamente nel momento in cui la CPU lo sta leggendo. Notate che le cinque linee d'indirizzamento superiori del codice dispositivo, da A3 ad A7, non vengono usate in nessun punto di questo circuito. Perciò, tutti i codici dispositivo seguenti "decodificano" nello stesso modo per attivare il canale di uscita $\overline{01}$:

01, 09, 11, 19, 21, 29, 31, 39, 41, 49 . . . . E1, E9, F1 ed F9

Ciò significa che il dispositivo d'ingresso collegato alla linea uno è indirizzato da trentadue codici dispositivo diversi . . . il che non è catastrofico, ma spesso indesiderato. Quindi, si dice che il suddetto circuito di decodifica, *decodifica in modo ambiguo* il codice dispositivo a otto bit, in quanto il dispositivo collegato ad una qualunque delle linee di uscita del 74LS138 non è univocamente indirizzato.

Esempio 2: Decodifica assoluta

Il circuito di Figura 3-12 decodifica *in modo assoluto* il codice dispositivo a otto bit, dato che ciascun bit di indirizzo da A0 ad A7 influenza l'impulso di selezione dispositivo che viene generato. La Figura 3-12 mostra, in generale, come è possibile generare uno qualunque dei 256 impulsi di selezione dispositivo collegando tre linee di uscita del decodificatore/demultiplexer agli ingressi di gate OR (per impulsi attivi sul basso) e gate NOR (per attivi sull'alto).
Ciascun impulso di selezione dispositivo è realizzato come uscita di un gate OR o NOR i cui ingressi sono le linee di uscita dei tre decodificatori/demultiplexer 74LS138. Per esempio, un impulso di selezione dispositivo attivo sul basso per il codice dispositivo A3 esadecimale, viene prodotto da un gate OR i cui ingressi sono:

Canale 3 del 74LS138 N. 1	(per 011 in A2-A0)	
Canale 4 del 74LS138 N. 2	(per 100 in A5-A3)	1010 0011 = A3H
Canale 2 del 74LS138 N. 3	(per 10 in A7-A6)	

Ci dobbiamo riferire ad un impulso di selezione dispositivo in base all'operazione a lui associata, IN (lettura di $\overline{I/O}$: $\overline{IORQ}$ e $\overline{RD}$ attivi) o $\overline{OUT}$ (scrittura di I/O, IORQ e WR attivi), ed al codice dispositivo come segue:

IN n o OUT n o $\overline{IN\ n}$ o $\overline{OUT\ n}$

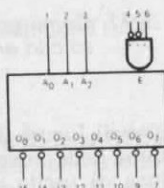
Tavola 3-1. Caratteristiche del T54LS138/T74LS138.

1-OF-8 DECODER/DEMULTIPLEXER

DESCRIPTION – The LSTTL/MSI T54LS138/T74LS138 is a high speed 1-of-8 Decoder/Demultiplexer. This device is ideally suited for high speed bipolar memory chip select address decoding. The multiple input enables allow parallel expansion to a 1-of-24 decoder using just three LS138 devices or to a 1-of-32 decoder using four LS138s and one inverter. The LS138 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- DEMULTIPLEXING CAPABILITY
- MULTIPLE INPUT ENABLE FOR EASY EXPANSION
- TYPICAL POWER DISSIPATION OF 32 mW
- ACTIVE LOW MUTUALLY EXCLUSIVE OUTPUTS
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

LOGIC SYMBOL

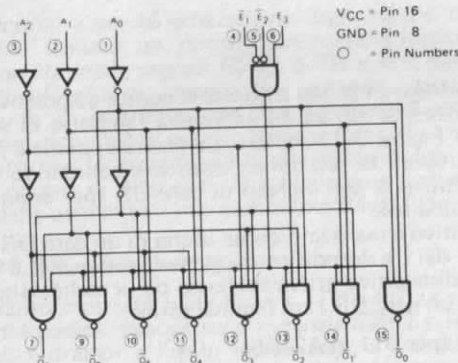


V_{CC} = Pin 16
GND = Pin 8

PIN NAMES		LOADING (Note a) HIGH	LOW
$A_0 - A_2$	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{E}_1, \overline{E}_2$	Enable (Active LOW) Inputs	0.5 U.L.	0.25 U.L.
E_3	Enable (Active HIGH) Input	0.5 U.L.	0.25 U.L.
$\overline{O}_0 - \overline{O}_7$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



CONNECTION DIAGRAM DIP (TOP VIEW)



Tavola 3-1. Caratteristiche del T54LS138/T74LS138 (seguito).

FUNCTIONAL DESCRIPTION – The LS138 is a high speed 1-of-8 Decoder/Demultiplexer fabricated with the low power Schottky barrier diode process. The decoder accepts three binary weighted inputs (A_0, A_1, A_2) and when enabled provides eight mutually exclusive active LOW outputs ($\overline{O}_0$-$\overline{O}_7$). The LS138 features three Enable inputs, two active LOW ($\overline{E}_1$, $\overline{E}_2$) and one active HIGH (E_3). All outputs will be HIGH unless $\overline{E}_1$ and $\overline{E}_2$ are LOW and E_3 is HIGH. This multiple enable function allows easy parallel expansion of the device to a 1-of-32 (5 lines to 32 lines) decoder with just four LS138s and one inverter. (See Figure a.)

The LS138 can be used as an 8-output demultiplexer by using one of the active LOW Enable inputs as the data input and the other Enable inputs as strobes. The Enable inputs which are not used must be permanently tied to their appropriate active HIGH or active LOW state.

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$	$\overline{O}_4$	$\overline{O}_5$	$\overline{O}_6$	$\overline{O}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

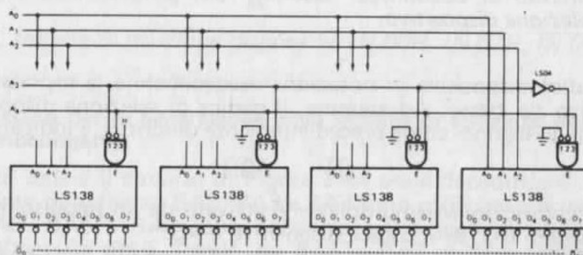


Fig. a

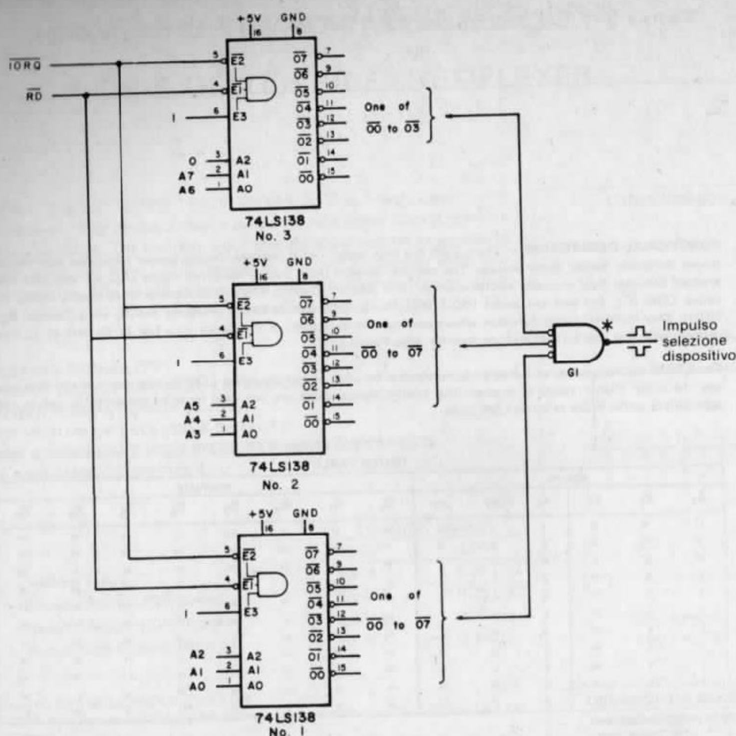


Figura 3-12. Circuito di decodifica "assoluta" per generare 256 diversi impulsi di selezione dispositivo.

dove n è il codice dispositivo in notazione esadecimale e la soprassegnatura indica un segnale attivo sul basso. Ad esempio, il codice di selezione dispositivo prodotto dal gate OR con gli ingressi come precedentemente descritto, è indicato da:

$$\overline{\text{IN}} \quad \overline{\text{A3H}}$$

In quanto il decodificatore/demultiplexer è attivato da un impulso negativo durante un ciclo di lettura di I/O, ed il codice dispositivo è A3.

La Figura 3-13 mostra il circuito con gate NOR ed OR per i codici di selezione dispositivo IN 00H, IN 01H, $\overline{\text{IN 02H}}$, $\overline{\text{IN A3H}}$.

Il suddetto circuito risulta eccessivo per la maggior parte delle applicazioni, dato che in genere non è necessario generare tutti i 256 impulsi di selezione dispositivo. In genere, gli impulsi di selezione dispositivo presenti in un sistema vengono definiti come una sequenza di indirizzi in alcuni sottoinsiemi dell'intero range da 00 a FF. Il Nanocomputer è in grado di generare una decodifica parziale degli impulsi di selezione dispositivo da 00 a 1F (Si veda la definizione dei tempi di $\overline{\text{IOQ } 0 \div 3}$, $\overline{\text{IOU } 0 \div 3}$, $\overline{\text{IOE } 0 \div 3}$ alla fine di questo capitolo).

Il circuito di Figura 3-11 e 3-12 è utile solo se gli impulsi di selezione dispositivo di un sistema vengono distribuiti casualmente nella gamma da 00 a FF. Se i codici dispositivo sono sequenziali, questo circuito non è da preferirsi.

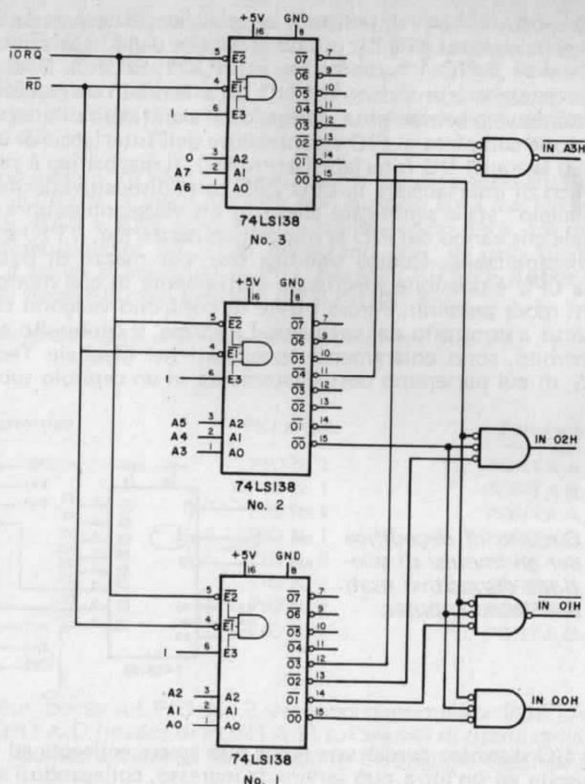


Figura 3-13. Impulsi di selezione dispositivo IN 00H, IN 01H, $\overline{\text{IN 02H}}$ e $\overline{\text{IN A3H}}$.

Esempio 3: Circuito per la generazione degli impulsi di selezione del dispositivo del Nanocomputer

Come potete vedere il circuito di Figura 3-14 usa il decodificatore/demultiplexer 74LS138. Le linee di indirizzo BA2, BA3 e BA4 sono collegate agli ingressi di selezione, mentre BA5, BA6 e BA7 sono collegate agli ingressi di abilitazione, con un invertitore sull'ingresso BA6. Perciò da BA5 a BA7 devono essere tutte a livello logico zero perchè il chip venga attivato. Questo esempio è diverso dai precedenti esempi di decodifica perchè i segnali $\overline{\text{IORQ}}$ e $\overline{\text{WR}}$ o $\overline{\text{RD}}$ *non entrano affatto in gioco per attivare il decodificatore.* I segnali $\overline{\text{RD}}$ e $\overline{\text{WR}}$ vengono usati insieme ad un impulso inviato sulle linee di uscita del decodificatore/demultiplexer, allo scopo di attivare i dispositivi PIO. Questo circuito illustra pertanto un'altra tecnica di decodifica, che consiste nel generare un impulso strettamente dipendente dal codice dispositivo, e poi nell'usare $\overline{\text{IORQ}}$ e $\overline{\text{RD}}$ o $\overline{\text{WR}}$ per determinare se l'operazione di IN o di OUT. Questa strategia di decodifica serve per limitare il numero di decodificatore/demultiplexer che sono necessari per generare tutti gli impulsi di selezione dispositivo e/o di selezione della memoria, dato che nessun decodificatore/demultiplexer viene impiegato per eseguire una sola delle operazioni $\overline{\text{IORQ}}$, $\overline{\text{MEMR}}$, $\overline{\text{RD}}$ o $\overline{\text{WR}}$. Notate che, negli esempi precedenti, i chip 74LS138 venivano attivati solo per le operazioni di IN.

Dato che le linee BA0 e BA1 di indirizzo vengono poste in ingresso al PIO, questo circuito decodifica in assoluto gli 8 bit più bassi del bus degli indirizzi.

Come potete vedere il PIO 1 ha come ingressi $\overline{IOQ1}$, $\overline{BIORQ}$, BA0 e BA1. Senza entrare troppo in dettaglio a proposito del PIO (lo si vedrà in un capitolo successivo), vogliamo ora presentarvelo brevemente, spiegando il significato di questi ingressi.

La denominazione completa di PIO è controllore dell'Interfaccia di Ingresso/Uscita Parallela dello Z80 (Parallel I/O Interface Controller). Il dispositivo è stato progettato allo scopo specifico di interfacciare la CPU Z80 con i dispositivi periferici. La terminologia "I/O parallelo" sta a significare che otto bit viaggiano sempre in parallelo su otto linee parallele che vanno dal PIO al dispositivo periferico. Il PIO è un dispositivo a due porte, programmabile. Questo significa che, per mezzo di byte di controllo provenienti dalla CPU è possibile specificare esattamente in che modo il dispositivo opera, fra i molti modi possibili. Perciò i byte di controllo vengono creati e posti in uscita sul PIO sotto il controllo del software. La forma, il contenuto ed il significato dei byte di controllo, sono chiaramente specificati nel Manuale Tecnico del PIO, della SGS-ATES, di cui parleremo dettagliatamente in un capitolo successivo. Il PIO

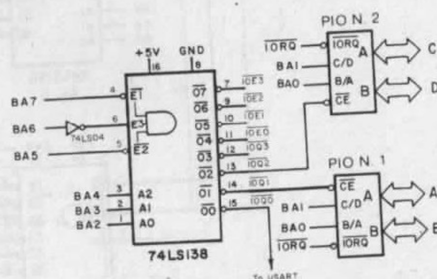


Figura 3-14. Circuito di decodifica per gli impulsi di selezione dispositivo usati dal Nanocomputer.

ha due porte di I/O distinte: perciò una porta può essere collegata ad una stampante come porta di uscita ed un'altra può servire da ingresso, collegandosi ad una tastiera. Queste due porte si chiamano PORTA A e PORTA B. Su ciascun PIO ognuna di queste porte presenta una configurazione indipendente tramite i propri byte di controllo.

Sul PIO vi sono quindi due porte di controllo, una per la porta A ed una per la porta B. Riassumendo il PIO è collegato in tutto con quattro porte (PORTA A, PORTA B e le associate porte di controllo), perciò, quando la CPU Z80 si interfaccia con il PIO, è necessario specificare la porta desiderata.

Questa è la ragione per cui le due linee d'indirizzo meno significative vengono decodificate dal PIO nel suddetto circuito di Figura 3-14. Cosicchè una linea seleziona C/D (Controllo o Dati). Inoltre, il PIO deve essere attivato soltanto per operazioni di I/O per le quali sono state indirizzate le sue porte. Ciò avviene, nella Figura 3-14, per mezzo dei segnali $\overline{BIORQ}$, e $\overline{IOQ1}$. Riassumiamo ora le funzioni di ognuno dei segnali $\overline{BIORQ}$, $\overline{IOQ1}$, BA0 e BA1:

$\overline{BIORQ}$: Questo segnale indica che è attiva in quel momento un'operazione di I/O. Chiaramente questo è un prerequisito per l'attivazione del PIO.

$\overline{IOQ1}$: Questo segnale è collegato al $\overline{CE}$, abilitazione del chip, pin del PIO N. 1. $\overline{IOQ1}$ è attivato solo se A2 è a livello logico uno e da A3 ad A7 sono tutti a livello logico zero. $\overline{IOQ1}$ sta ad indicare quindi una decodifica delle sei linee alte d'indirizzo, che ha lo scopo di specificare quale PIO dovrebbe essere attivo, mentre A0 e A1 specificano quale delle quattro porte viene in quel momento indirizzata, la PORTA A, il controllo della PORTA A, la PORTA B, o il controllo della PORTA B.

BA0,BA1: Questi segnali specificano quale porta sul PIO, indirizzata dalle linee da BA2 a BA7, dovrebbe essere attiva durante questa operazione. La porta A o B viene selezionata dalla linea BA0, la quale se è a livello logico uno seleziona la Porta B. L'I/O di controllo o dei dati è selezionato dalla linea BA1, BA1 a livello logico uno seleziona il controllo.

Per quanto riguarda il PIO N. 2, le sue linee d'ingresso sono $\overline{BIORQ}$, $\overline{IOQ2}$, BA0 e BA1. Quindi il PIO N. 2 viene attivato solo quando BA3 è a livello logico uno e BA2, BA4, BA5, BA6 e BA7 sono tutti a livello logico zero. Le uscite $\overline{IOQn}$ del decodificatore/demultiplexer 74LS138 assicurano che, in caso di qualunque operazione di I/O, è attivo al massimo un PIO. I segnali $\overline{BIORQ}$, BA0 e BA1 eseguono per il PIO N. 2 le stesse funzioni elencate per il PIO N. 1.

Eccovi ora l'elenco degli indirizzi delle porte dei due PIO interfacciati con il Nanocomputer:

Codice dispositivo	PIO attivo	Porta sul PIO attivo
04	PIO N. 1	PORTA A, dati
05	PIO N. 1	PORTA B, dati
06	PIO N. 1	PORTA A, controllo
07	PIO N. 1	PORTA B, controllo
08	PIO N. 2	PORTA C, dati
09	PIO N. 2	PORTA D, dati
0A	PIO N. 2	PORTA C, controllo
0B	PIO N. 2	PORTA D, controllo

Notate che le due porte sul PIO N. 2 vengono denominate PORTA C (invece di PORTA A) e PORTA D (invece di PORTA B) sui segnali di uscita della vostra scheda per esperimenti. Questo avviene al solo scopo di distinguere facilmente le porte dei due PIO.

Non abbiamo parlato della correlazione esistente fra i PIO e i segnali $\overline{WR}$ e $\overline{RD}$. Come fa il PIO a distinguere fra le operazioni d'ingresso e quelle di uscita? Per ora vi assicuriamo che è in grado di farlo, rimandando la discussione al riguardo al capitolo dedicato al dispositivo PIO.

Il Nanocomputer usa un circuito molto simile a questo per eseguire l'I/O con due PIO. Per la versione più versatile del Nanocomputer e per i microcomputer CLZ80, l'$\overline{IOQ0}$ seleziona un USART 8251 (ricevitore/trasmettitore sincrono/asincrono universale), della INTEL.

I restanti $\overline{IOQ3}$, $\overline{IOE0}$, $\overline{IOE1}$, $\overline{IOE2}$ e $\overline{IOE3}$ sono a vostra disposizione perchè possiate usarli come volete.

Si veda il Socket B, pin 31-35 sulla scheda per esperimenti del Nanocomputer. Si noti che il Socket B possiede altri segnali di decodifica disponibili:

$\overline{IOU0}$, $\overline{IOU1}$, $\overline{IOU2}$ e $\overline{IOU3}$

Questi segnali sono generati con un decodificatore/demultiplexer 74LS139 che genera anche quattro impulsi di selezione indirizzo, come da Figura 3-15. In questo esempio abbiamo descritto un circuito realizzato realmente nel Nanocomputer, quindi i nomi dei segnali usati non sono le semplici uscite (come D0-D7, A0-A15) dalla CPU Z80, ma i segnali bufferizzati come BA0, BA15. La bufferizzazione è

richiesta in quanto la CPU Z80 non ha un fan-out sufficiente per la struttura dei bus e per i decodificatori.

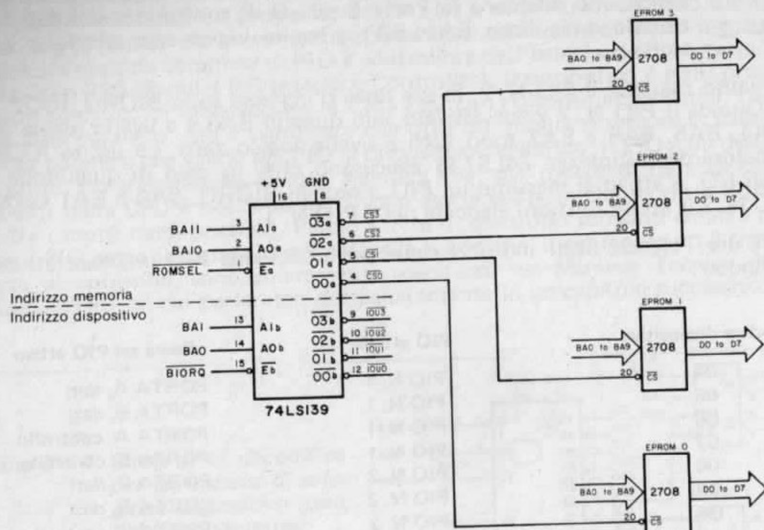


Figura 3-15. Circuito simile a quello usato dal Nanocomputer per generare i segnali $\overline{IOU0}$, $\overline{IOU1}$, $\overline{IOU2}$ e $\overline{IOU3}$ e per decodificare gli indirizzi situati nei 4K byte superiori della memoria.

Impulsi di selezione indirizzi

I circuiti che generano gli impulsi di selezione indirizzo sono molto simili a quelli che generano gli impulsi di selezione dispositivo. Vi sono però due differenze fondamentali:

1. Vengono decodificate sedici linee d'indirizzo, anzichè otto, e $\overline{MREQ}$ sostituisce $\overline{IORQ}$ come attivatore primario del circuito di decodifica.

2. I dispositivi di memoria eseguono da soli la maggior parte della decodifica degli indirizzi.

Gli esempi che seguono mostrano il circuito di decodifica del Nanocomputer, che decodifica appunto indirizzi nei 4K byte di memoria EPROM da F000 a FFFF. Notate che questa memoria è costituita da quattro EPROM 2708, e che ogni EPROM memorizza i dati relativi ai seguenti indirizzi di memoria:

Indirizzi	Zoccoli corrispondenti	
EPROM N. 1 : F000 – F3FF	Q49	(LHS)
EPROM N. 2 : F400 – F7FF	Q50	
EPROM N. 3 : F800 – FBFF	Q51	
EPROM N. 4 : FC00 – FFFF	Q52	(RHS)

Se riscrivete la serie di indirizzi suddetti in codice binario, potete vedere che i bit A10 e A11 identificano in modo univoco il range in cui un dato indirizzo rientra:

	BA15				BA0
EPROM N. 1 :	1111	00	00	0000	0000
	1111	00	11	1111	1111
EPROM N. 2 :	1111	01	00	0000	0000
	1111	01	11	1111	1111
EPROM N. 3 :	1111	10	00	0000	0000
	1111	10	11	1111	1111
EPROM N. 4 :	1111	11	00	0000	0000
	1111	11	11	1111	1111

Quindi, per esempio, l'indirizzo FCCC = 1111 1100 1100 1100 ha sia A10 che A11 a livello logico uno e, pertanto, risiede nella EPROM N. 4: l'indirizzo F503=1111 0101 0000 0011 ha A10 a livello logico uno e A11 a livello logico zero e risiede perciò nella EPROM N. 2. Vediamo ora come queste informazioni vengono usate dal Nanocomputer per decodificare gli indirizzi di memoria a sedici bit sul bus degli indirizzi, allo scopo di attivare, quando è il caso, la EPROM 2708 esatta.

Esempio 1: Impulsi di selezione indirizzi per la memoria EPROM del Nanocomputer

Il circuito nella Figura 3-15 usa un doppio decodificatore/demultiplexer 74LS139 da 2 a 4 linee. Questo dispositivo ha due ingressi di strobe, due coppie di ingressi di selezione e due gruppi di quattro linee di uscita.
Di seguito sono fornite nella Tavola 3-2 alcune informazioni sul 74LS139, tratte dal Low-Power Schottky TTL ICs Data Book della SGS-ATES, per vostra comodità. La Figura 3-15 mostra come il Nanocomputer usi entrambi i decodificatori/demultiplexer sul chip:

- Il decodificatore/demultiplexer "a" seleziona una delle 4 EPROM 2708
- Il decodificatore/demultiplexer "b" produce i segnali $\overline{IOU0}$, $\overline{IOU1}$, $\overline{IOU2}$, $\overline{IOU3}$

Studieremo in dettaglio la funzione del decodificatore/demultiplexer "a".

Il segnale $\overline{ROMSEL}$ (ROM SELect) che attiva il decodificatore "a" è generato dal circuito di Figura 3-16.

Una definizione semplificata ma logicamente equivalente di $\overline{ROMSEL}$ è la seguente:

$$ROMSEL = \overline{A12 \cdot A13 \cdot A14 \cdot A15 \cdot MREQ \cdot \overline{RFSH}}$$

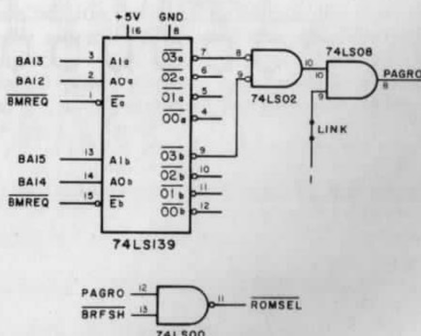


Figura 3-16. Circuito usato dal Nanocomputer per generare gli impulsi di selezione $\overline{ROM}$, $\overline{ROMSEL}$ e PAGRO (dove $\overline{BRFSH}$ è il segnale $\overline{RFSH}$ della CPU Z80, bufferizzato).

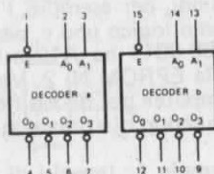
Tavola 3-2. Caratteristiche del T54LS139/T74LS139.

DUAL 1-OF-4 DECODER

DESCRIPTION — The LSTTL/MSI T54LS139/T74LS139 is a high speed Dual 1-of-4 Decoder/Demultiplexer. The device has two independent decoders, each accepting two inputs and providing four mutually exclusive active LOW outputs. Each decoder has an active LOW Enable input which can be used as a data input for a 4-output demultiplexer. Each half of the LS139 can be used as a function generator providing all four minterms of two variables. The LS139 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- SCHOTTKY PROCESS FOR HIGH SPEED
- MULTIFUNCTION CAPABILITY
- TWO COMPLETELY INDEPENDENT 1-OF-4 DECODERS
- ACTIVE LOW MUTUALLY EXCLUSIVE OUTPUTS
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

LOGIC SYMBOL

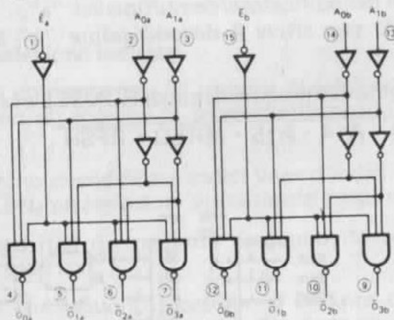


V_{CC} = Pin 16
GND = Pin 8

PIN NAMES		LOADING (Note a) HIGH	LOW
A_0, A_1	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{E}$	Enable (Active LOW) Input	0.5 U.L.	0.25 U.L.
$\overline{O}_0 - \overline{O}_3$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



V_{CC} = Pin 16
GND = Pin 8
○ = Pin Numbers

CONNECTION DIAGRAM DIP (TOP VIEW)

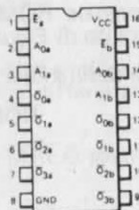


Tavola 3-2. Caratteristiche del T54LS139/T74LS139 (seguito).

FUNCTIONAL DESCRIPTION – The LS139 is a high speed dual 1-of-4 decoder/demultiplexer fabricated with the Schottky barrier diode process. The device has two independent decoders, each of which accept two binary weighted inputs (A_0, A_1) and provide four mutually exclusive active LOW outputs ($\overline{O}_0$-$\overline{O}_3$). Each decoder has an active LOW Enable ($\overline{E}$). When $\overline{E}$ is HIGH all outputs are forced HIGH. The enable can be used as the data input for a 4-output demultiplexer application.

Each half of the LS139 generates all four minterms of two variables. These four minterms are useful in some applications, replacing multiple gate functions as shown in Fig. a, and thereby reducing the number of packages required in a logic network.

TRUTH TABLE

INPUTS			OUTPUTS			
$\overline{E}$	A_0	A_1	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

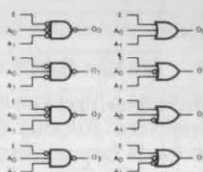
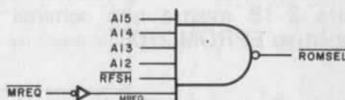


Fig. a

Questo segnale può essere generato da un gate NAND singolo, come dato in Figura 3-17, rappresentazione logica della precedente relazione.

Figura 3-17. Realizzazione del circuito di decodifica di Figura 3-16 che genera il segnale $\overline{ROMSEL}$, con un solo gate.



$\overline{ROMSEL}$ è attivo solo e solo A12, A13, A14 e A15 sono tutti ad 1 logico, $\overline{MREQ}$ è attivo (a 0 logico) e $\overline{RFSH}$ non è attivo. Si noti che $\overline{ROMSEL}$ è il NAND di PAGRO e $\overline{RFSH}$. Allora, perchè $\overline{ROMSEL}$ sia attivo sul basso, $\overline{RFSH}$ *non* deve essere attivo (deve, ad esempio, essere ad 1) e PAGRO deve essere (ad 1 logico). La presenza di RFSH assicura che non si realizza l'accesso ad una delle EPROM quando è in atto un ciclo di refresh della memoria. L'uso del segnale RFSH nell'ambito della logica di decodifica è un mezzo per essere sicuri che i bus sono "puliti" prima di una qualsiasi operazione di lettura di dati della EPROM. Vedrete nel Capitolo 4 perchè sia necessaria questa pulizia del bus. Anche per il segnale PAGRO occorre analizzare quando è attivo (1 logico); riferendosi alla Figura 3-16, è:

PAGRO = 1 se e solo se (entrambi gli ingressi al gate AND del 74LS08 sono 1)

PAGRO = 1 se e solo se (l'uscita dal NOR del 74LS02 è 1)

PAGRO = 1 se e solo se (entrambi gli ingressi al NOR del 74LS02 sono 0)

PAGRO = 1 se e solo se (le uscite $\overline{O3}$a e $\overline{O3}$b del 74LS139 sono 0)

Dato che: $\overline{O3a}$ = se e solo se $\overline{BMREQ}$ è attivo AND (BA12 = 1) AND (BA13 = 1)

$\overline{O3b}$ = se e solo se $\overline{BMREQ}$ è attivo AND (BA14 = 1) AND (BA15 = 1)

Alla fine è:

PAGRO = 1 se e solo se ($\overline{BMREQ}$ = 0 AND BA12 = 1 AND BA13 = 1 AND BA14 = 1 AND BA15 = 1)

Allora, dato che $\overline{BMREQ}$ = 0 è equivalente a BMREQ = 1,

$$PAGRO = BA12 \cdot BA13 \cdot BA14 \cdot BA15 \cdot BMREQ$$

e quindi:

$$ROMSEL = \overline{BA12 \cdot BA13 \cdot BA14 \cdot BA15 \cdot BMREQ \cdot \overline{BRFSH}}$$

Il decodificatore/demultiplexer è attivo solo se è in atto una operazione di accesso in memoria e l'indirizzo sul bus degli indirizzi cade nel range F000–FFFF. La sola distinzione ancora da fare consiste nell'individuare quale EPROM deve essere attivata. Questo si realizza nella Figura 3-15 usando BA10 e BA11 per selezionare una delle quattro uscite del 74LS139, $\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$ e $\overline{CS3}$. Ogni segnale CS_n (chip select n) è collegato alla EPROM associata con l'appropriata decodifica BA10 e BA11.

Abbiamo cioè descritto il ruolo per le linee di indirizzo di A10 e A15 nell'indirizzamento di un byte di EPROM. Cosa c'è da dire per le linee da A0 ad A9? Esse sono collegate come ingressi (da BA0 e BA9 in Figura 3-15) a ciascuna delle quattro EPROM.
Dato che ciascuna EPROM 2708 contiene esattamente 1024, o 2^{10} byte, dieci linee (da A0 ad A9) possono specificare quale byte, di ogni EPROM, deve essere posto sul bus dei dati. La EPROM stessa associa uno dei suoi byte con l'indirizzo presente alle dieci linee di ingresso, e pone il byte sulle sue otto linee di uscita. La Figura 3-18 mostra uno schema a blocchi e la configurazione dei pin per il dispositivo EPROM 2708.

Esempio 2: Impulsi di selezione indirizzi per la memoria RAM del Nanocomputer

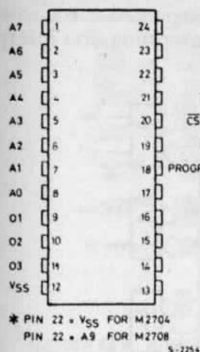
Teniamo a dire, all'inizio di questo esempio, che introdurremo tutta una serie di concetti leggermente avanzati. Se da una parte vi raccomandiamo di sforzarvi di capire il più possibile, dall'altra vi esortiamo a non scoraggiarvi nel caso in cui troviate delle difficoltà per voi insormontabili, almeno per ora.

Se non capite tutto ora, proseguite e ripassate questa parte in un secondo tempo. La Figura 3-19 mostra uno schema semplificato del circuito usato dal Nanocomputer per la decodifica degli impulsi di selezione degli indirizzi della memoria RAM. La memoria RAM standard del Nanocomputer è costituita da 8 dispositivi 4027 di RAM dinamica da 4 kilobit indirizzata nella zona di memoria da 0000 – 1000 (in esadecimale).

La locazione della memoria può essere variata modificando i ponticelli sulla scheda del Nanocomputer, ma, assumiamo, a proposito di questo specifico esempio, che i cavallotti siano posizionati in modo da localizzare la memoria nella regione di memoria 0000 – 1000 (in esadecimale).

La memoria RAM del Nanocomputer è strutturata in modo tale che ciascun chip fisico mette a disposizione un bit per ciascuno dei 4 kilobyte in memoria.
Al fine di ottenere un indirizzamento univoco dei 4 kilobit in ogni dispositivo RAM, sono necessarie 12 linee di indirizzo. Per evitare di avere 12 piedini di indirizzo in ingresso su ciascun dispositivo RAM, il costruttore lo ha progettato con solo sei pin di ingresso di indirizzo, in modo da accettare l'indirizzo in due momenti diversi, sei bit alla volta, da A0 ad A5 dapprima, come Row Address Select (RAS) (Selezione

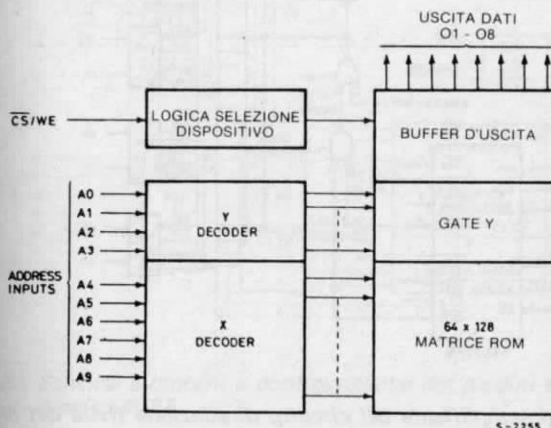
COLLEGAMENTI DEI PIEDINI



DESCRIZIONE PIEDINI

A_0-A_9	ADDRESS INPUTS (INGRESSI INDIRIZZI)
O_1-O_8	DATA OUTPUTS (USCITE DATI)
$\overline{CS}$/WE	CHIP SELECT/WRITE ENABLE INPUT (SELEZIONE DISPOSITIVO/ABILITAZIONE SCRITTURA)

SCHEMA A BLOCCHI



MODO	NUMERO PIEDINO						
	9,11,13,17	12	18	19	20	21	24
LETTURA	D_{OUT}	V_{SS}	V_{SS}	V_{DD}	V_{IL}	V_{BB}	V_{CC}
PROGRAMMAZIONE	D_{IN}	V_{SS}	Pulsed V_{IHP}	V_{DD}	V_{IHW}	V_{BB}	V_{CC}

Figura 3-18. Schema a blocchi e configurazione dei piedini del dispositivo EPROM 2708.

di Riga), e da A6 ad A11 dopo come Column Address Select (CAS) (Selezione di Colonna). Per la configurazione dei piedini del dispositivo della RAM dinamica 4027, si veda la Figura 3-20.

Si nota che due piedini sono chiamati $\overline{RAS}$ e $\overline{CAS}$. Quando il segnale $\overline{RAS}$ è attivo (basso), i sei bit di indirizzo sulle linee MA0-MA5 sono interpretati come row address (indirizzo di riga), mentre un segnale $\overline{CAS}$ attivo (basso) determina una interpretazione dell'indirizzo come indirizzo di colonna.

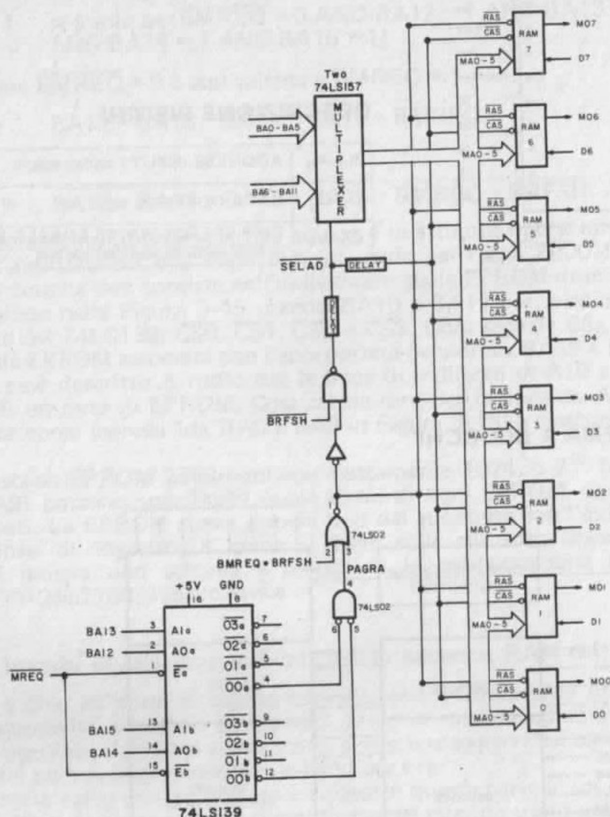
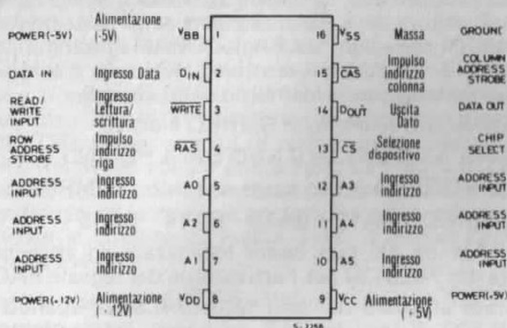


Figura 3-19. Illustrazione semplificata del circuito di selezione RAM del Nanocomputer.

Dato che i 4 kilobyte sono organizzati nel chip RAM come matrice $2^6 \times 2^6$, gli indirizzi riga e colonna determinano in modo univoco la cella di memoria (un bit) cui accedere.

L'utilizzo dei sei ingressi di indirizzo per dodici bit di indirizzo a disposizione in due gruppi di sei bit viene indicato con il nome di *address per multiplexing*. Questa notazione è dovuta sia alla logica sofisticata all'interno del chip, sia quella presente all'esterno del dispositivo, con riferimento ai circuiti che lo interfacciano con la CPU. Il circuito utilizzato dal Nanocomputer è rappresentato dettagliatamente nella appendice. La Figura 3-19 mostra i segnali importanti, dando un sommario dei

COLLEGAMENTI DEI PIEDINI



SCHEMA A BLOCCHI

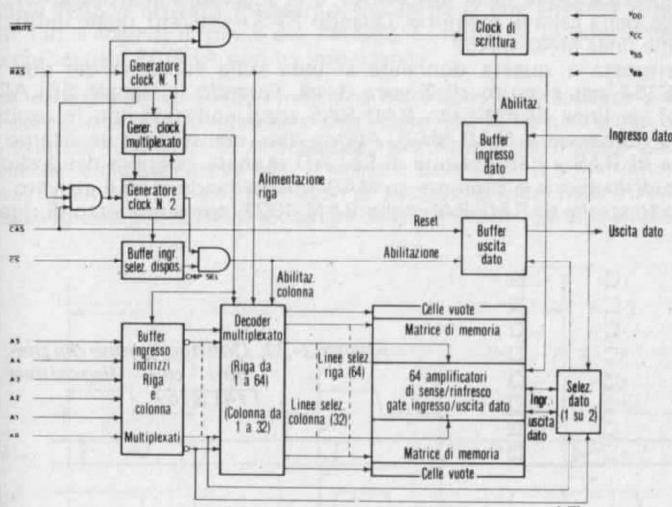


Figura 3-20. Schema a blocchi e configurazione dei piedini del dispositivo RAM dinamica 4027.

circuiti per la generazione dei segnali $\overline{\text{SELAD}}$ e $\overline{\text{CAS}}$, con una "function box". Durante una lettura o scrittura in memoria si realizzano i seguenti eventi:

1. L'indirizzo della locazione di memoria è posto sul bus degli indirizzi da BA0 a BA15.

2. BA12, BA13, BA14 e BA15 sono posti in ingresso al decodificatore/demultiplexer doppio a 2 linee a 4 linee 74LS139.

3. Viene attivato il segnale $\overline{\text{BMREQ}}$, con conseguente abilitazione di entrambi i decodificatori/demultiplexer del chip 74LS139.

4. Se BA12, BA13, BA14 e BA15 sono tutti a livello logico 0, questo vuol dire che si sta indirizzando una locazione di memoria nella regione 0000-1000 (in esadecimale). Allora le RAM dinamiche sono selezionate dalla attivazione del segnale PAGRA, che significa PAge of RAM (pagina di RAM). Con riferimento alla Figura 3-19, PAGRA è attivo (alto) solo e solo se 00a e 00b sono attivi (basso) sul decodificatore/demultiplexer. Quindi,

   PAGRA è attivo alto se e solo se $\overline{\text{BMREQ}}$ è attivo

   AND BA12 = 0 AND BA13 = 0 AND BA14 = 0 AND BA15 = 0

   Allora, dato che $\overline{\text{BMREQ}}$ è attivo basso se e solo se BMREQ è alto:

   PAGRA = BMREQ · BA12 · BA13 · BA14 · BA15

   La memoria RAM da 4K può essere localizzata ad altri indirizzi scegliendo differenti uscite del 74LS139 per l'attivazione del segnale PAGRA.

5. PAGRA è inviato al pin 3 del gate NOR 74LS02. Quando PAGRA va alto, l'uscita del 74LS02, il segnale $\overline{\text{RAS}}$, va basso, indipendentemente dal valore logico dell'altro ingresso del gate NOR.

6. L'attivazione del segnale $\overline{\text{RAS}}$ determina la lettura da parte del chip RAM 4027, dell'indirizzo sulle linee MA0-MA5, e la successiva interpretazione di indirizzo di riga della cella di memoria. Quando $\overline{\text{RAS}}$ è attivato, quale indirizzo è presente sulle linee MA0-MA5?
   La risposta a questa domanda si basa sulla funzione dei due multiplexer 74LS157 nel circuito di Figura 3-19. Quando il segnale SELAD è inattivo (alto), le linee di indirizzo BA0-BA5 sono collegate con le uscite del multiplexer connesse a MA0-MA5. Allora, dato che sussiste un ritardo tra l'attivazione di $\overline{\text{RAS}}$ e l'attivazione di $\overline{\text{SELAD}}$ le metà inferiore dei dodici bit di indirizzo di memoria è presente su MA0-MA5 quando $\overline{\text{RAS}}$ è attivato. Allora $\overline{\text{RAS}}$ attua lo strobe di BA0-BA5 nelle RAM 4027, come indirizzo di riga.

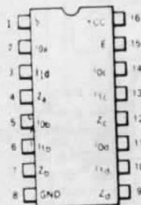


Figura 3-21. Configurazione dei piedini del dispositivo 74LS157.

7. Dopo un ritardo, il segnale RAS attivo basso fa si che SELAD vada basso. Quando questo si verifica, viene attuato il collegamento di BA6-BA11 sulle sue linee di uscita. Quindi, dopo un altro ritardo, BA6-BA11 si stabilizzano su MA0-MA5, al che il segnale CAS è messo basso.

8. Con l'attivazione di CAS, la metà superiore dei dodici bit di indirizzo di memoria, BA6-BA11, viene sottoposta a strobe nelle RAM 4027 come indirizzo di colonna.

Si noti che nella precedente sequenza di eventi, il segnale $\overline{\text{BRFSH}}$ non è stato menzionato, in quanto esso non gioca un ruolo significativo nelle operazioni di lettura e scrittura in memoria.
Allora perchè il segnale $\overline{\text{BRFSH}}$ appare in questo circuito? La risposta a questa domanda può essere data solo dopo una necessaria trattazione di una importante proprietà delle memorie RAM dinamiche: la necessità di un *refresh* (rinfresco o riscrittura) periodico.

Le RAM dinamiche realizzano la memorizzazione dei bit mediante la carica di una capacità, invece di posizionare un flip-flop. Quindi si rende necessario un "refresh" (rinfresco) periodico della memoria prima che il condensatore si scarichi. La CPU Z80 fornisce segnali di refresh alla RAM dinamica in termini di un indirizzo di refresh ed un impulso di strobe di refresh indicato con il nome di $\overline{\text{RFSH}}$. Questo viene bufferizzato e si presenta sul bus del Nanocomputer come $\overline{\text{BRFSH}}$. L'indirizzo di refresh è un indirizzo ad 8 bit di selezione riga e colonna memorizzato nel registro R interno alla CPU Z80. Questi segnali di refresh sono generati dalla CPU al termine di ciascun ciclo M1 (fetch del codice operativo). La Figura 3-22 mostra un ciclo M1 della CPU Z80. Si noti che in corrispondenza con l'attivazione di $\overline{\text{RFSH}}$, il contenuto del registro R (l'indirizzo di refresh), è posto sulla metà inferiore del bus degli indirizzi. Con riferimento al circuito di Figura 3-19, il refresh delle RAM dinamiche è realizzato dai seguenti eventi:

1. L'attivazione del segnale $\overline{\text{BRSFH}}$ inibisce i segnali $\overline{\text{SELAD}}$ e $\overline{\text{CAS}}$ in quanto $\overline{\text{SELAD}}$ è forzato alto.
2. I primi 6 bit dell'indirizzo di refresh su BA0-5 sono successivamente abilitati dai multiplexer 74LS157 verso gli ingressi di RAM MA0-MA5.
3. Il segnale RAS è attivato in quanto sia BMREQ che BRSFH sono entrambi attivati (ad 1 logico) al pin 2 del 74LS02. Si noti che in questo caso lo stato logico del segnale PAGRA non ha importanza.
4. Con la attivazione di $\overline{\text{RAS}}$, l'indirizzo di refresh è sottoposto a clock nella RAM come indirizzo di selezione riga. Dato che $\overline{\text{CAS}}$ non è mai abilitato, la RAM utilizza l'indirizzo di refresh per ricaricare le celle nella specifica riga. Con ciascun ciclo successivo M1, il registro di refresh R si incrementa. Allora, successivi cicli M1 attuano il refresh di successive righe nella RAM dinamica, interessando una intera RAM dinamica di 4Kx8 bit in 128 cicli di refresh.

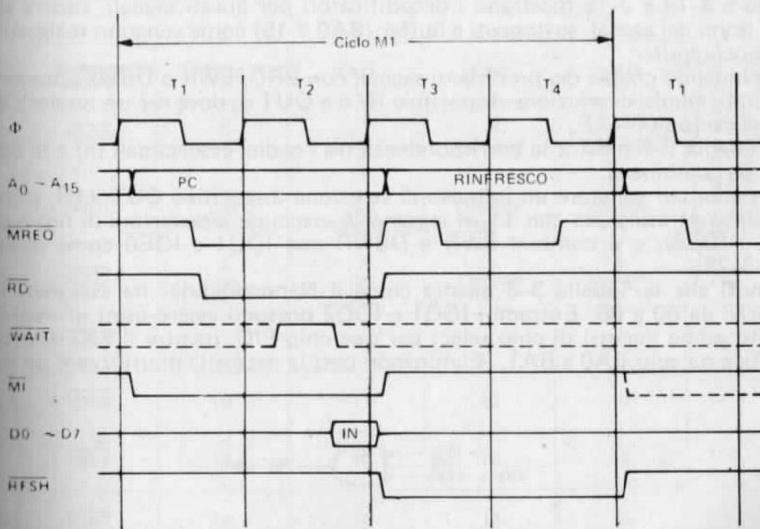


Figura 3-22. Ciclo M1 della CPU Z80.

SOMMARIO DEGLI IMPULSI DI SELEZIONE DI I/O DISPONIBILI SULLA BREADBOARDING STATION DEL NANOCOMPUTER

Come abbiamo precedentemente indicato, vi sono nove segnali di selezione di I/O, IOQ3, $\overline{IOE0}$, $\overline{IOE1}$, $\overline{IOE2}$, $\overline{IOE3}$, $\overline{IOU0}$, $\overline{IOU1}$, $\overline{IOU2}$ e $\overline{IOU3}$, resi disponibili sullo zoccolo B della scheda per esperimenti del Nanocomputer. In questa sezione eseguiamo una revisione di come vengono definiti e discutiamo su come usarli in circuiti di I/O che richiedono che gli otto bit inferiori del bus degli indirizzi siano decodificati in modo non ambiguo.

In totale, vi sono 12 segnali di selezione di I/O generati sulle schede del Nanocomputer.
I 9 resi disponibili sulla piastra per esperimenti più $\overline{IOQ0}$, $\overline{IOQ1}$ ed $\overline{IOQ2}$. Gli ultimi tre segnali sono per uso esclusivo del sistema operativo del Nanocomputer e non sono quindi disponibili all'utente. Per completezza diamo di seguito la definizione di tutti i 12 segnali:

$$
\begin{aligned}
IOU0 &= \overline{BIORQ} \cdot \overline{BA0} \cdot \overline{BA1} \\
IOU1 &= \overline{BIORQ} \cdot BA0 \cdot \overline{BA1} \\
IOU2 &= \overline{BIORQ} \cdot \overline{BA0} \cdot BA1 \\
IOU3 &= \overline{BIORQ} \cdot BA0 \cdot BA1 \\
IOQ0 &= \overline{BA2} \cdot \overline{BA3} \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOQ1 &= BA2 \cdot \overline{BA3} \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOQ2 &= \overline{BA2} \cdot BA3 \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOQ3 &= BA2 \cdot BA3 \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOE0 &= \overline{BA2} \cdot \overline{BA3} \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOE1 &= BA2 \cdot \overline{BA3} \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOE2 &= \overline{BA2} \cdot BA3 \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7} \\
IOE3 &= BA2 \cdot BA3 \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}
\end{aligned}
$$

Le Figure 3-14 e 3-15 mostrano i decodificatori per questi segnali; ancora abbiamo usato i nomi dei segnali sottoposti a buffer (BA0 ÷ 15) come vengono realmente usati nel Nanocomputer.

Combinando coppie dei precedenti segnali con $\overline{BRD}$, $\overline{BWR}$ o $\overline{DBWR}$, possono esse-re generati impulsi di selezione dispositivo IN n e OUT n, dove n è un numero esadeci-male nel campo 00 - 1F.

La Tabella 3-3 mostra la corrispondenza tra i codici esadecimali (n) e le coppie di segnali da combinarsi.
Ad esempio, per generare un impulso di selezione dispositivo OUT 11H, si individui la casella contrassegnata con 11, si leggano le associate intestazioni di riga e colonna ($\overline{IOU1}$ e $\overline{IOE0}$), e si combini $\overline{BWR}$ o $\overline{DBWR}$ con $\overline{IOU1}$ e $\overline{IOE0}$ come indicato in Figura 3-23.

Si noti che la Tabella 3-3 mostra come il Nanocomputer ha assegnati i codic dispositivi da 00 a 0B. Entrambi $\overline{IOQ1}$ e $\overline{IOQ2}$ possono essere usati in modo molto adeguato come ingressi di chip select per due chip PIO, mentre il PIO direttament decodifica da solo BA0 e BA1. (Eliminando così la necessità di utilizzare un qualsias

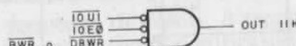


Figura 3-23. Esempio di combinazione dei segnali di selezione dispositivo dell Tabella 3-3.

segnale $\overline{\text{IOUn}}$ per l'indirizzamento del PIO). Quando realizzerete da soli un circuito per il PIO in un successivo esperimento, sarete in grado di fare un buon uso, nello stesso modo, di $\overline{\text{IOQ3}}$ (o di qualsiasi segnale IOEn).

ALTRI USI DEGLI IMPULSI DI SELEZIONE DISPOSITIVO
La sostituzione dell'hardware con il software

Gli impulsi di selezione dispositivo sono semplici da generare e per nulla costosi. In una tipica applicazione, un decodificatore/demultiplexer 74LS138 costa circa $1,25 ovvero più o meno 16 cent per ognuno degli otto impulsi di selezione dispositivo che è in grado di generare. Gli impulsi di selezione dispositivo vengono usati in molte applicazioni, oltre che per l'accesso in memoria o per l'invio di impulsi di strobe ai dispositivi di I/O. In questo paragrafo, parleremo di alcune delle applicazioni più comuni di tali impulsi, che illustrano interessanti evoluzioni dall'hardware al software.

Esempio 1: Uso di impulsi di selezione dispositivo per generare impulsi di clock

E' facile scrivere un programma che generi un solo impulso negativo:

```
OUT     (1AH),A
```

Per una CPU a 2,5 MHz, l'ampiezza di questo impulso (come l'AND di IORQ e di WR), è di circa 2,5 cicli di 1,0 microsecondi. Per una sequenza di impulsi di

Tabella 3-3. I segnali di selezione dispositivo del Nanocomputer. I segnali da $\overline{\text{IOU0}}$ a $\overline{\text{IOU3}}$ sono generati dalle linee di indirizzo A0 ed A1, mentre quelli da $\overline{\text{IOQ0}}$ a $\overline{\text{IOE3}}$ sono generati dalle linee da A2 ad A4.

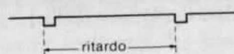
Indirizzo di I/O	IOU0	IOU1	IOU2	IOU3
$\overline{\text{IOQ0}}$	0 UART	1 UART	2 Not used	3 I/O Flag
$\overline{\text{IOQ1}}$	4 Dati Porta "A"	5 Dati Porta "B"	6 Controllo Porta "A"	7 Controllo Porta "B"
$\overline{\text{IOQ2}}$	8 Dati Porta "C"	9 Dati Porta "D"	A Controllo Porta "C"	B Controllo Porta "D"
$\overline{\text{IOQ3}}$	C	D	E	F
$\overline{\text{IOE0}}$	10	11	12	13
$\overline{\text{IOE1}}$	14	15	16	17
$\overline{\text{IOE2}}$	18	19	1A	1B
$\overline{\text{IOE3}}$	1C	1D	1E	1F

(Righe da $\overline{\text{IOQ0}}$ a $\overline{\text{IOQ2}}$:) DECODER PER USO INTERNO NON UTILIZZABILE ESTERNAMENTE

(Righe da $\overline{\text{IOQ3}}$ a $\overline{\text{IOE3}}$:) LINEE UTILIZZABILI

questo tipo, possiamo usare questo semplice programma:

```
PULSE:  OUT (1AH),A
        CALL DELAY
        JP PULSE
```



dove la subroutine DELAY è un loop di ritardo programmato eseguito nell'intervallo di tempo che intercorre fra un impulso e l'altro. Questi impulsi dovrebbero essere presenti come impulsi di selezione dispositivo di indirizzo 1AH. E per variare l'ampiezza dell'impulso? Lo si può fare facilmente, usando un flip-flop 74LS74, o altro analogo, ed una coppia di impulsi di selezione dispositivo, uno per l'ingresso di set ed uno per l'ingresso di clear del flip-flop (Figura 3-24).

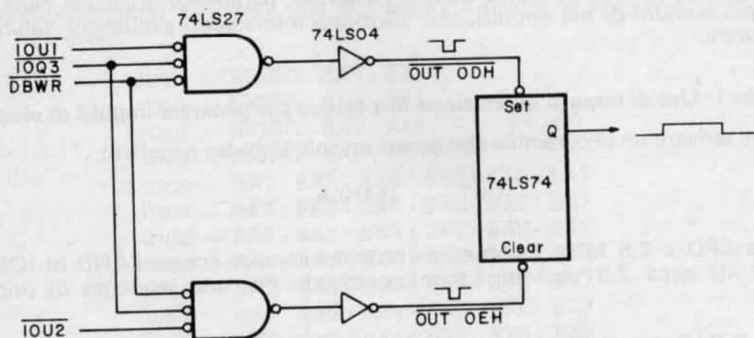


Figura 3-24. Come generare da software impulsi di durata variabile.

Con due loop di ritardo temporizzati, è possibile generare un flusso di impulsi con u duty cycle noto (Figura 3-25).
Questi ultimi due circuiti sostituiscono un multivibratore monostabile o astabil (come il timer 555 utilizzato come un oscillatore).

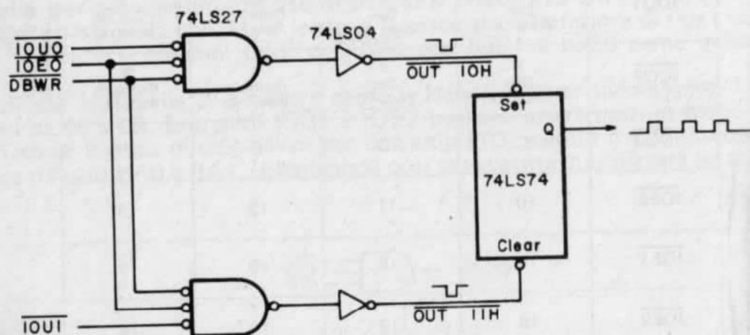


Figura 3-25. Come generare da software dei treni di impulsi.

Esempio 2: Invio di impulsi di strobe a circuiti integrati

Un importante applicazione dei microcomputer consiste nell'inviare impulsi di strobe a singoli circuiti integrati presenti in strumenti dispositivi elettronici. Per esempio, tali impulsi possono:

- Azzerare contatori, shift register, flip-flop e latch;
- Caricare contatori, shift register e latch;
- Abilitare multiplexer, demultiplexer, decodificatori, selettori di dati, contatori, shift register, memorie, codificatori di priorità, UART, PIO, e moltissimi altri dispositivi;
- Inibire gli ingressi di clock di contatori e di shift register;
- Azzerare, commutare e temporizzare flip-flop;
- Selezionare le funzioni di scorrimento a sinistra, di scorrimento a destra, di caricamento e di inibizione in shift register;
- Interrompere altri elaboratori digitali.

Usare gli impulsi di selezione dispositivi per controllare le operazioni di singoli circuiti integrati, significa sostituire l'hardware con il software.

Spesso un obiettivo della progettazione circuitale è quello di minimizzare il numero dei componenti. Indubbiamente, l'impiego dei microcomputer permette al progettista di sostituire con il software le funzioni normalmente implementate da hardware, e questo può portare ad una riduzione del numero dei dispositivi. Pur essendo la sostituzione dell'hardware con il software spesso molto desiderabile, esistono delle condizioni in base alle quali questo non è realizzabile. Innanzitutto, mentre il costo dell'hardware va generalmente diminuendo, il costo dello sviluppo del software è in costante aumento.

E' possibile che il costo di un programma che sostituisce un circuito hardware, anche se ammortizzato su centinaia o migliaia di unità, non sia inferiore, a livello singola unità, al suo equivalente hardware. In secondo luogo, il software è spesso più lento dell'hardware, perciò bisogna prendere in considerazione tutte le esigenze di velocità e di temporizzazione. Ricordate, un gate AND può avere un ritardo di propagazione che va da 10 a 100 nanosecondi, mentre un'istruzione AND viene eseguita in 4 cicli-T (circa 1,6 microsecondi, sul Nanocomputer).

RIEPILOGO

Per ognuna delle seguenti istruzioni, indicate quanti cicli di fetch del codice operativo dalla memoria (M1), di lettura in memoria, di scrittura in memoria, di ingresso e di uscita dei dispositivi, vengono eseguiti durante l'esecuzione dell'istruzione. Per esempio, l'istruzione LD A,B necessita della esecuzione di un ciclo di fetch dalla memoria da parte della CPU, e di niente altro, mentre l'istruzione INC (IX) richiede l'esecuzione, da parte della CPU di:

2 cicli di fetch dalla memoria del codice operativo (M1), perchè l'istruzione ha un codice operativo a 2 byte;

1 ciclo di ingresso in memoria per caricare il byte all'indirizzo puntato da IX;

1 ciclo di scrittura in memoria per caricare di nuovo il byte incrementato nello indirizzo puntato da IX.

Basandovi sugli esempi suddetti, completate la tabella che segue.

Istruzione	Cicli M1	Cicli di lettura in memoria	Cicli di scrittura	Cicli di ingresso	Cicli di uscita
LD A,B	1	0	0	0	0
INC (IX)	2	1	1	0	0
LD A,00H					
LD C,D					
INC A					
DEC (HL)					
LD HL,00FFH					
LD (BC),A					
LD (0010H),A					
POP HL					
PUSH BC					
LD (0100H), 1234H					
EXX					
XOR A					
XOR (HL)					
DAA					
DEC IX					
RRA					
RLC (IY + 03)					
SET 3,(HL)					
DJNZ F4H					
JP (IX)					
CALL 0100H					
RET					
IN A,(04H)					
OUT (05H),A					
IN B,(C)					
OUT (C), D					

RISPOSTA

Istruzione	Cicli M1	Cicli di lettura in memoria	Cicli di scrittura	Cicli di ingresso	Cicli di uscita
LD A,B	1	0	0	0	0
INC (IX)	2	1	1	0	0
LD A,00H	1	1	0	0	0
LD C,D	1	0	0	0	0
INC A	1	0	0	0	0
DEC (HL)	1	1	1	0	0
LD HL,00FH	1	2	0	0	0
LD (BC),A	1	0	1	0	0
LD (0010H),A	2	2	1	0	0
POP HL	1	2	0	0	0
PUSH BC	1	0	2	0	0
LD (0100H),1234H	2	2	2	0	0
EXX	1	0	0	0	0
XOR A	1	0	0	0	0
XOR (HL)	1	1	0	0	0
DAA	1	0	0	0	0
DEC IX	2	0	0	0	0
RRA	1	0	0	0	0
RLC (IY + 03)	2	1	1	0	0
SET 3,(HL)	2	1	1	0	0

DJNZ F4H	1	1	0	0	0
JP (IX)	2	0	0	0	0
CALL 0100H	1	2	2	0	0
RET	1	2	0	0	0
IN A,(04H)	1	0	0	1	0
OUT (05H),A	1	0	0	0	1
IN B,(C)	1	0	0	1	0
OUT (C),D	1	0	0	0	1

INTRODUZIONE AGLI ESPERIMENTI

Gli esperimenti che seguono dimostrano come è possibile generare ed usare gli impulsi di selezione dispositivo. Eseguiremo anche il montaggio di un circuito di "cattura" dei bit che vi permetterà di osservare i singoli bit sul bus degli indirizzi e sul bus dei dati mentre la CPU esegue le istruzioni di accesso alla memoria e/o di I/O.

Esperimento N.	Commento
1	Illustra un circuito di "cattura dei bit" che vi permette di campionare i bit individuali sul bus degli indirizzi e sul bus dei dati. Userete questo circuito per osservare i bus suddetti durante l'esecuzione delle istruzioni di accesso alla memoria e di I/O.
2	Mostra un circuito di decodifica che usa un decodificatore/demultiplexer 74LS139.
3	Illustra l'uso degli impulsi di selezione dispositivi per inviare impulsi di strobe ad un contatore a decade 7490.
4	Illustra un circuito di memoria che utilizza due chip Intel di memoria statica 2101 256x4 ed una porta AND a 4 ingressi per la decodifica degli indirizzi.
5	Illustra la differenza tra memoria statica e dinamica con riferimento all'effetto degli stati di attesa sulle operazioni di refresh della memoria della CPU.

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è costruire un circuito di "cattura dei bit" che vi permetterà di osservare il bus degli indirizzi e il bus dei dati durante l'esecuzione delle istruzioni di accesso alla memoria e di I/O.

Schema del circuito (Figura 3-26)

Programma LOOP1

Codice oggetto	Codice sorgente	Commento
D3 C5	LOOP1: OUT (0C5H),A	; Poni in uscita il contenuto dell'ac- ; cumulatore sulla porta C5
18 FC	JR LOOP1	; Ripeti fino al reset

Configurazioni dei pin dei circuiti integrati (Tavole 3–3 e 3–4)

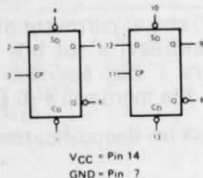
Tavola 3-3. Caratteristiche del T54LS74/T74LS74.

DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

DESCRIPTION – The T54LS74/T74LS74 dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and $\overline{Q}$ outputs.

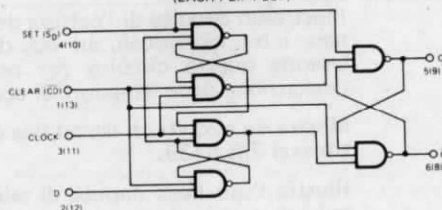
Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.

LOGIC SYMBOL



V_{CC} = Pin 14
GND = Pin 7

LOGIC DIAGRAM (EACH FLIP-FLOP)



MODE SELECT – TRUTH TABLE

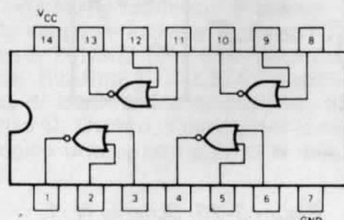
OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{S}_D$	$\overline{C}_D$	D	Q	$\overline{Q}$
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

*Both outputs will be HIGH while both $\overline{S}_D$ and $\overline{C}_D$ are LOW, but the output states are unpredictable if $\overline{S}_D$ and $\overline{C}_D$ go HIGH simultaneously.

H,h = HIGH Voltage Level
L,l = LOW Voltage Level
X = Don't Care
l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

Tavola 3-4. Caratteristiche del T54LS02/T74LS02.

QUAD 2-INPUT NOR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS02X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS02X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.6	3.2	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			2.4	5.4	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 273 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES

1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

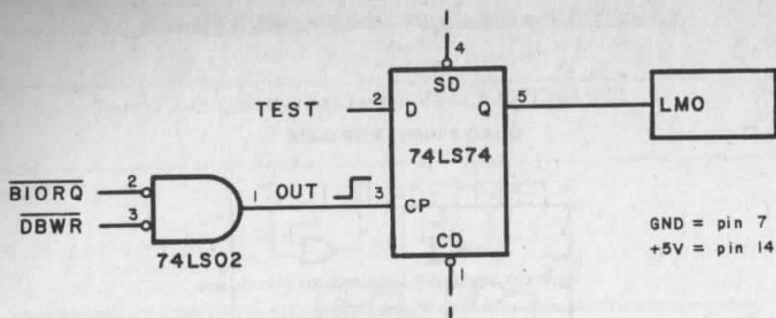


Figura 3-26. Schema N. 1.

Passo 1

Montate il circuito di Figura 3-26. Si noti che la porta 74LS02 nello schema precedente non è rappresentata usando l'usuale simbolo di porta NOR; mostrato in Figura 3-27A, ma con il simbolo rappresentato in Figura 3-27B.

I piccoli circoletti all'ingresso della Porta AND sono interpretati come equivalenti di invertitori. Allora, l'uscita della Porta AND è un 1 logico se e solo se entrambi gli ingressi sono 0 logici. Si noti che questo comportamento è esattamente quello di una Porta NOR. Quindi, i due simboli precedenti rappresentano una funzione logica "NOR". Discuteremo delle rappresentazioni alternative della porta più dettagliatamente nel Capitolo 4.

L'ingresso "D" del circuito integrato 74LS74 dovrebbe essere inizialmente collegato alla linea BA7 del bus degli indirizzi.

Questo è realizzato collegando il filo di *"test"* allo zoccolo C, Pin 29. Contrassegnato BA7, sulla scheda per esperimenti del Nanocomputer. Come già detto, BA7 che sta per Buffered Address, è stato collegato allo zoccolo C tramite un buffer. BA7 corrisponde ad un "TEST" point nello schema.

L'indicatore luminoso LM0 sarà acceso se il TEST point è a stato logico 1 il flip-flop è sottoposto a clock, altrimenti sarà spento.

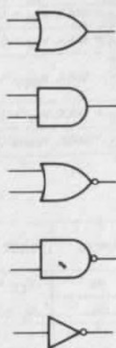


Figura 3-27A.

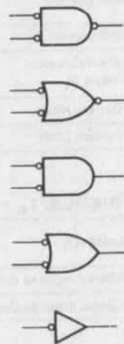


Figura 3-27B.

Passo 2

Caricate il programma LOOP1 del Nanocomputer ed iniziatene l'esecuzione. L'indicatore luminoso è acceso o spento?

Dovreste osservare che l'indicatore luminoso è acceso. Durante l'esecuzione della istruzione OUT (0C5H),A, vengono attivate le linee $\overline{IORQ}$ e $\overline{WR}$ della CPU Z80, in modo che l'uscita del gate 74LS02 (che è un impulso di selezione di dispositivo di uscita) và all'ingresso del flip-flop D 74LS74 "positive edge triggered". Quando il flip-flop sul suo ingresso di clock il fronte positivo, il segnale sul suo ingresso D viene copiato sulla sua uscita Q. Ovvero, l'indicatore luminoso sarà acceso se il punto di "TEST" è a livello logico uno, e sarà spento se tale punto è a livello logico zero. (Si veda la Figura 3-28).

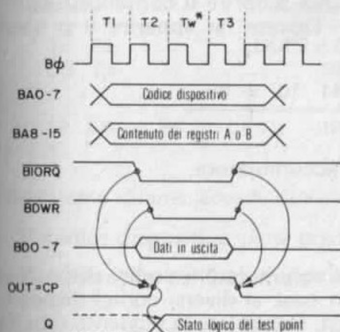


Figura 3-28. Diagramma di temporizzazione per la cattura dei bit durante un'operazione di uscita.

Potete ora capire perchè il circuito si chiama di *"cattura dei bit"*. Il circuito è attualmente configurato per *"catturare"* il valore logico sulla linea BA7 del bus degli indirizzi durante un ciclo di uscita. Riepiloghiamo quanto avviene sul bus degli indirizzi e sul bus dei dati durante un ciclo di uscita:

1. Il codice dispositivo (in questo caso C5) viene posto sulla metà inferiore del bus degli indirizzi, BA0-BA7.
2. Il contenuto dell'accumulatore viene posto sulla r. tà superiore del bus degli indirizzi, BA8-BA15.
3. Il contenuto dell'accumulatore viene posto sul bus dei dati.

Dato che il codice dispositivo è C5, le linee da BA0 a BA7, durante il ciclo di uscita generato dall'istruzione OUT (0C5H),A, dovrebbe risultare:

BA –	7	6	5	4	3	2	1	0
	1	1	0	0	0	1	0	1

Passo 3

Senza interrompere l'esecuzione del programma, controllate tutte le linee del bus dei dati registratene qui i risultati:

BD –	7	6	5	4	3	2	1	0

(Ricordate che BDn sono le linee dati Dn della CPU Z80 sottoposte a buffer).
Quale pensate che sia il contenuto dell'accumulatore?

Per verificare la vostra ipotesi, premete il tasto BREAK sulla tastiera del Nanocomputer e spostate l'indicatore luminoso in posizione AF. (Ricordate che, se premete il tasto RESET, il valore dell'accumulatore, nonchè quello degli altri registri, è inizializzato a 00, mentre, con la pressione di BREAK, i registri non vengono modificati). Dovreste osservare che il contenuto del Registro A, visualizzato sulla tastiera del Nanocomputer, corrisponde a quanto avete visto sul bus dei dati con il vostro circuito di "cattura dei bit".

Passo 4

Ora usate il vostro circuito di cattura dei bit per scoprire il contenuto delle otto linee alte d'indirizzo (NON DIMENTICATE – Dovete far apparire il programma LOOP1 a LOOP1)

BA – 15 14 13 12 11 10 9 8

Avreste dovuto osservare anche il contenuto dell'accumulatore.

Passo 5

Facciamo ora degli esperimenti ponendo nell'accumulatore codici dispositivo diversi e diversi byte. Modificate il programma in base ai diversi codici dispositivo e caricate l'accumulatore con i differenti byte di test. Compilate la seguente tabella:

Codice dispositivo	Accumulatore	BA-15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0	CC																
C1	1C																

Dovreste osservare una netta corrispondenza fra il codice dispositivo e BA0-BA7, e fra il contenuto dell'accumulatore a BA8-BA15.

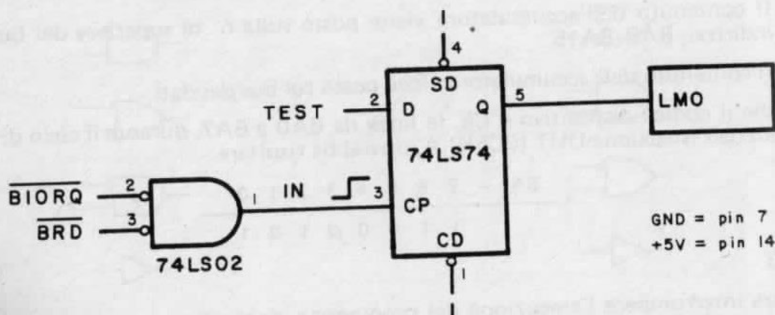


Figura 3-29. Schema N. 2.

Passo 6

Catturiamo ora alcuni bit durante un ciclo d'ingresso. Apportereste una lieve modifica al vostro circuito di cattura dei bit?

E' sufficiente cambiare l'ingresso al gate NAND da $\overline{DBWR}$ in $\overline{BRD}$. Il nuovo circuito si presenta come in Figura 3-29.
Dobbiamo apportare una modifica anche al programma, usando il programma LOOP2, in modo che generi cicli di INGRESSO anzichè cicli di uscita. Il motivo per cui si inizializza l'accumulatore prima dell'esecuzione dell'istruzione IN, sarà spiegato in seguito.

Programma LOOP2

Codice oggetto	Codice sorgente	Commenti
3E 21	LOOP2: LDA,21H	; Inizializza l'accumulatore
DB C5	IN A,(0C5H)	; Carica un byte di dati dalla Porta ; C5
18 FA	JR LOOP2	; Ripeti fino al reset

Riepiloghiamo quanto accade durante un ciclo d'ingresso:

1. Il codice dispositivo viene posto sulla metà inferiore del bus degli indirizzi.
2. Il contenuto dell'accumulatore viene posto sulla metà superiore del bus degli indirizzi.
3. I segnali $\overline{IORQ}$ e $\overline{RD}$ vengono attivati, ed è generato un impulso selezione dispositivo in ingresso.
4. La CPU legge sul bus dei dati un byte d'ingresso posto sul bus stesso da un dispositivo periferico, in risposta ai segnali suddetti.

Eseguite il programma precedente a partire dalla locazione LOOP2.
I punti 1, 2 e 3 possono essere facilmente verificati eseguendo degli esperimenti che sfruttano il nuovo arrangiamento del circuito di cattura dei bit. Come potete vedere dalla Figura 3-30, campioniamo il bus dei dati nel momento in cui supponiamo che

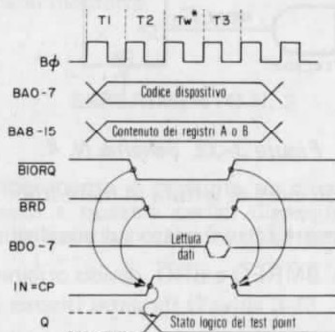


Figura 3-30. Diagramma dei tempi per la cattura dei bit durante un'operazione di ingresso.

il dispositivo periferico d'ingresso stia ponendo i suoi dati affinchè la CPU li legga. Dato che alla Porta C5 non è stato collegato nessun dispositivo d'ingresso, non possiamo controllare che cosa accade sul bus dei dati durante il ciclo d'ingresso. Perciò, un modo per controllare il punto 4, di cui sopra, sarebbe stato quello di collegare un dispositivo d'ingresso alla Porta C5, controllare che cosa pone sul bus di dati durante un ciclo d'ingresso ed osservare se quel byte d'ingresso veniva realmente letto nell'accumulatore. In questo esperimento, dovreste osservare che il valore FF è "letto" nell'accumulatore come risultato dell'istruzione di input. Procedete a Single Step con il programma per verificare questo effetto nei registri AF. Inizializziamo l'accumulatore all'inizio di ogni loop in modo da poter osservare il cambiamento del suo contenuto.

Passo 7

Per "catturare" i bit durante l'accesso in memoria, dovete pensare una modifica del "bit catcher circuit" come segue:

PER LA LETTURA IN MEMORIA (Figura 3-31)

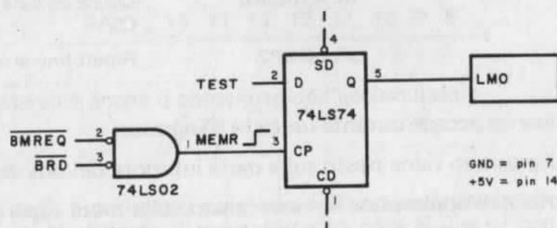


Figura 3-31. Schema N. 3.

PER LA SCRITTURA IN MEMORIA (Figura 3-32)

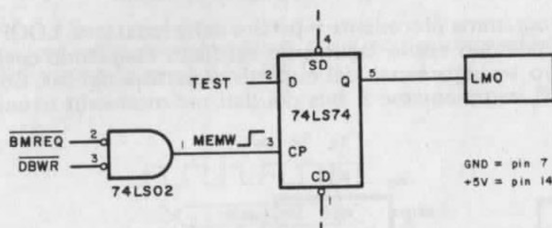


Figura 3-32. Schema N. 4.

Ricordiamo cosa accade in un ciclo di lettura in memoria:

1. L'indirizzo che deve essere fatto è posto sul bus degli indirizzi.
2. Sono attivati i segnali $\overline{\text{BMREQ}}$ e $\overline{\text{BRD}}$, dando origine ad un segnale di indirizzo MEMR
3. La locazione di memoria indirizzata pone il suo contenuto sul bus dei dati
4. La CPU legge il contenuto della locazione di memoria dal bus dei dati.

Sussiste senz'altro una notevole somiglianza tra l'accesso in memoria e l'I/O di un dispositivo periferico. *Tuttavia*, vi è una differenza che influenza in modo critico il circuito di cattura dei bit. Il "bit catcher", come definito per la cattura di byte di I/O, è attivato durante *qualsiasi* ciclo di ingresso o di uscita. Analogamente lo schema n. 3 Figura 3-31 definisce che è attivato durante qualsiasi ciclo di lettura in memoria, e lo schema n. 4 Figura 3-32 definisce un circuito che è attivato durante *qualsiasi* ciclo di scrittura in memoria. Con queste chiavi di lettura, potete individuare perchè non possiamo usare il circuito di cattura dei bit dello schema n. 3 Figura 3-31 per catturare gli indirizzi ed i byte in ingresso per le visualizzazioni di lettura in memoria con LD A,(HL)?

La ragione è questa: durante l'esecuzione del programma, ogni istruzione inizia con un ciclo M1, o ciclo di fetch. Un ciclo M1 non è altro che un ciclo di lettura in memoria con il segnale $\overline{M1}$ attivato! Quindi all'esecuzione del programma, ogni istruzione coinvolge almeno un ciclo di lettura in memoria, che attiva il "bit catcher". Allora, durante l'esecuzione di un loop infinito come quello sopra indicato, il circuito di "cattura dei bit" non conserverà i bit di indirizzo o di dati, in quanto essi varieranno costantemente.

Sarebbe allora d'aiuto inserire della logica nello schema n. 3 Figura 3-31 per impedire che il circuito di cattura dei bit subisca impulsi di trigger dai cicli M1?

La risposta è NO. Il ciclo M1 si verifica solo nel corso del fetch dalla memoria del codice operativo. Molte istruzioni contengono byte diversi dai codici operativi (es. LD A,<B2>).

Questi byte, che non sono codici operativi vengono prelevati dalla memoria per mezzo di cicli di lettura della memoria stessa, e con il segnale $\overline{M1}$ *non* attivo.
Quindi, dobbiamo per forza concludere che il nostro circuito di cattura dei bit, così com'è, è inadeguato, per lo meno per catturare i dati e gli indirizzi delle letture in memoria. Per risolvere questo problema, è possibile apportare al circuito molte modifiche. Lasciamo questo compito ai progettisti digitali in erba che ci stanno leggendo. Potreste anche prendere in considerazione il problema di catturare i dati e gli indirizzi delle scritture in memoria.

ESPERIMENTO N. 2

Scopo

Lo scopo di questo esperimento è costruire un circuito che decodifichi quattro codici dispositivo d'ingresso e quattro codici dispositivo di uscita. Controllerete in modo parziale il circuito usando il circuito di cattura dei bit che avete montato nell'Esperimento 1.

Configurazioni dei pin dei circuiti integrati (Tavola 3-5)

Schema del circuito (Figura 3-33)

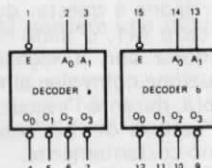
Tavola 3-5. Caratteristiche del T54LS139/T74LS139.

DUAL 1-OF-4 DECODER

DESCRIPTION – The LSTTL/MSI T54LS139/T74LS139 is a high speed Dual 1-of-4 Decoder/Demultiplexer. The device has two independent decoders, each accepting two inputs and providing four mutually exclusive active LOW outputs. Each decoder has an active LOW Enable input which can be used as a data input for a 4-output demultiplexer. Each half of the LS139 can be used as a function generator providing all four minterms of two variables. The LS139 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- **SCHOTTKY PROCESS FOR HIGH SPEED**
- **MULTIFUNCTION CAPABILITY**
- **TWO COMPLETELY INDEPENDENT 1-OF-4 DECODERS**
- **ACTIVE LOW MUTUALLY EXCLUSIVE OUTPUTS**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**

LOGIC SYMBOL



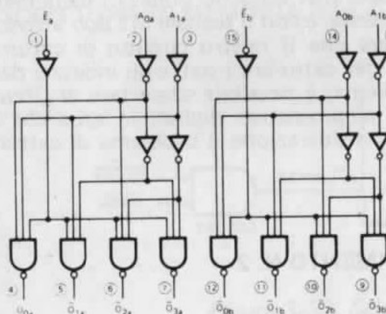
PIN NAMES

		LOADING (Note a) HIGH	LOW
A_0, A_1	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{E}$	Enable (Active LOW) Input	0.5 U.L.	0.25 U.L.
$\overline{O}_0 - \overline{O}_3$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

V_{CC} = Pin 16
GND = Pin 8

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



V_{CC} = Pin 16
GND = Pin 8
○ = Pin Numbers

CONNECTION DIAGRAM
DIP (TOP VIEW)

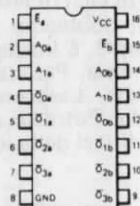


Tavola 3-5. Caratteristiche del T54LS139/T74LS139 (seguito).

FUNCTIONAL DESCRIPTION – The LS139 is a high speed dual 1-of-4 decoder/demultiplexer fabricated with the Schottky barrier diode process. The device has two independent decoders, each of which accept two binary weighted inputs (A_0, A_1) and provide four mutually exclusive active LOW outputs ($\overline{O}_0$-$\overline{O}_3$). Each decoder has an active LOW Enable ($\overline{E}$). When $\overline{E}$ is HIGH all outputs are forced HIGH. The enable can be used as the data input for a 4-output demultiplexer application.

Each half of the LS139 generates all four minterms of two variables. These four minterms are useful in some applications, replacing multiple gate functions as shown in Fig. a, and thereby reducing the number of packages required in a logic network.

TRUTH TABLE

INPUTS			OUTPUTS			
$\overline{E}$	A_0	A_1	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

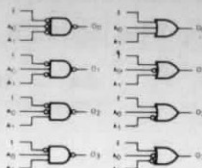


Fig. a

Programma DECODIFICA

Codice oggetto	Codice sorgente		Commenti
0E 20	DECODE:	LD C,20H	; Carica il codice dispositivo nel re- ; gistro C
06 C5		LD B,0C5H	; Carica il byte C5 nel registro B ; per poterlo osservare in seguito ; sulla metà superiore del bus d'in- ; dirizzo
ED 61	LOOP3:	OUT (C),H	; Poni in uscita il contenuto del re- ; gistro H sulla porta puntata regi- ; stro C
18 FC		JR LOOP3	; Ripeti l'istruzione di uscita fino al ; reset

Passo 1

Montate il circuito di decodifica mostrato nello schema precedente (Figura 3-33). Se non avete il circuito "bit catcher" fatto precedentemente per il LATCH di un singolo bit in un ciclo di uscita, per favore, montatelo; potete fare riferimento allo Esperimento n. 1 Figura 3-26.

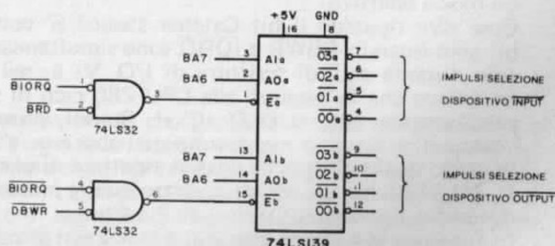


Figura 3-33. Schema N. 5.

Passo 2

Caricate ed eseguite il programma suddetto a partire da DECODE. Nel corso della esecuzione, usate il collegamento TEST del circuito di cattura dei bit per controllare tutti i bit del bus degli indirizzi. Scrivete di seguito i risultati:

BA –	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_

Avreste dovuto osservare quanto segue:

1. Il codice dispositivo, 20, è stato posto su BA0–BA7.
2. Il contenuto del registro B, C5, è apparso su BA8–BA15 sotto questa forma:

BA –	15	14	13	12	11	10	9	8
	1	1	0	0	0	1	0	1

Passo 3

Controllate tutti i bit del bus dei dati e riportate i risultati qui sotto:

BD –	7	6	5	4	3	2	1	0
	_	_	_	_	_	_	_	_

Dovreste essere in grado di dare il contenuto del registro H. Arrestate il programma, posizionate l'indicatore luminoso ad HL, e testate se i contenuti del registro H coincidono con i dati da noi osservati sul bus dei dati.
(ATTENZIONE: si usi il test BREAK al posto di quello di RESET per arrestare il programma, altrimenti correte il rischio di cambiare il contenuto dei reigstri!).
Reinizializzate il registro H a 21H ed eseguite ancora il programma. Dovreste osservare 21H sul bus dei dati, mentre il programma è in esecuzione.

Passo 4

Ricollegate gli ingressi di enable (abilitazione) $\overline{Ea}$ (pin1) ed $\overline{Eb}$ (pin 15) del decodificatore/demultiplexer 74LS139 a MASSA (GND), in modo che sia sempre abilitato. Idealmente tutto quello che vi serve consiste nel collegare il terminale TEST del circuito di cattura dei bit ai pin di uscita del 74LS139. Analizziamo la situazione:

Come prima cosa, dovete essere certi che il bit Catcher stà ancora tenendo sotto osservazione la giusta istruzione, OUT (C),H. Entrambi i decodificator/demultiplexer nel 74LS139 sono costantemente attivi, quindi essi non si comportano in un modo selettivo.
Cosa dire riguardo il bit Catcher stesso? E' configurato in modo da "catturare" bit solo quando $\overline{DBWR}$ e $\overline{IORQ}$ sono simultaneamente attivi, ma questo si verifica solo durante cicli di scrittura di I/O. Vi è, nel precedente programma, qualche istruzione che fa eseguire alla CPU Z80 cicli di scrittura di I/O? La risposta è SI, precisamente una, la OUT (C),H. Quindi, potete concludere che il bit Catcher "cattura" i bit solo mentre questa istruzione è in esecuzione. Ora, dovete essere in grado di anticipare che cosa vi aspettate di osservare.
Il 74LS139 ha un set di quattro uscite normalmente alte, un set per ciascun decodificatore/demultiplexer da 2 a 4 linee.
In funzione del valore logico ai due ingressi di select-- A0a (pin 2) ed A1a (pin 3),

per il decodificatore/demultiplexer "a" e A0b (pin 14) e A1b (pin 13) per il decodificatore/demultiplexer "b"-- uno di ciascun set di quattro uscite andrà basso durante il ciclo di scrittura di I/O generato dalla istruzione OUT (C),H. Quale? Dato che voi conoscete il contenuto del bus degli indirizzi, sapete la risposta. BA0-BA7 contiene il codice dispositivo presente nel registro, cioè 20. Quindi sia BA6 che BA7 sono a livello logico zero durante la esecuzione di OUT (C),H. Così potete anticipare che $\overline{O0a}$ (pin 4) e $\overline{O0b}$ (pin 12) saranno bassi. Le altre uscite del 74LS139 (pin 5, 6, 7, 9, 10 ed 11) saranno alte.

In sintesi, vi aspettate di leggere quanto segue alle uscite del decodificatore:

Pin 4 : zero logico	Pin 12 : zero logico
Pin 5 : uno logico	Pin 11 : uno logico
Pin 6 : uno logico	Pin 10 : uno logico
Pin 7 : uno logico	Pin 9 : uno logico

Utilizzate il "bit Catcher" (configurato per i cicli di uscite) per verificare questa anticipazione.

Passo 5

Ora cercate, con i seguenti codici dispositivo, di determinare il loro effetto sulle uscite del decodificatore.

40 60 80 C0

Per quanto riguarda i codici dispositivo 40 e 60, avreste dovuto osservare un livello logico zero sul pin 11; per il codice dispositivo 80, i pin 6 e 10 avrebbero dovuto abbassarsi, e per il codice dispositivo C0, avrebbero dovuto abbassarsi il pin 7 e 9. In ogni caso, dovrebbe essere stato esattamente un pin ad andare a livello logico zero, con tutti gli altri a livello logico uno. A questo punto, dovrebbe risultarvi ovvio che questo circuito di decodifica *non* decodifica in assoluto gli otto bit inferiori del bus degli indirizzi:

a. Vengono decodificate solo due linee, A6 e A7, e

b. Avete osservato che due diversi codici dispositivo (40 a 60) generano lo stesso impulso di selezione.

Passo 6

Montate di nuovo il circuito di cattura dei bit per i cicli d'ingresso (Esperimento 1, schema n. 2). Cambiate l'istruzione OUT (C),H del programma in IN H,(C), codice esadecimale ED 60. Ora vedete se le osservazioni che valevano per i cicli di scrittura di I/O valgono anche per i cicli di lettura di I/O. Lo potete fare variando il codice dispositivo come in precedenza e controllando gli effetti causati alle uscite del decodificatore/demultiplexer.

Come per i cicli di uscita, abbiamo osservato che il contenuto del registro B viene posto sugli otto bit più significativi del bus degli indirizzi, e che il contenuto del registro C è posto sugli otto bit meno significativi del bus degli indirizzi. Osservate che il bus dei dati non è significativo in questo passo, perchè non abbiamo controllato sul suo contenuto durante il ciclo di ingresso. Le uscite del decodificatore/demultiplexer mostrano lo stesso comportamento sia nei cicli di ingresso che di uscita.

Vale la pena di notare che la scelta dei codici dispositivo di prova non è stata dettata da un semplice capriccio. Abbiamo attentamente evitato la gamma da 00 a 1F per la quale il Nanocomputer genera i codici dispositivo.

Passo 7

Ricollegare gli ingressi di enable, $\overline{Ea}$ (pin 2) ed $\overline{Eb}$ (pin 15), del decodificatore/demultiplexer 74LS139 alle uscite delle due porte OR 74LS32, come indicato nello schema n. 5 Figura 3-33. Ora il decodificatore/demultiplexer "a" sarà attivato solo durante le operazioni di lettura di I/O (o IN), mentre il decodificatore/demultiplexer "b" sarà attivato durante le operazioni di scrittura di I/O (o OUT). Questo è un ottimo circuito di decodifica senz'altro valido. Potete a questo punto chiedervi perchè non lo abbiamo usato nei passi precedenti al posto di chiedervi di collegare a massa i pin $\overline{Ea}$ ed $\overline{Eb}$. Il motivo risiede nella *temporizzazione*. Ricordate che i passi precedenti facevano uso del circuito di cattura dei bit per poter osservare le uscite del 74LS139. Con i circuiti come da schema n. 5; (Figura 3-33) il bit catcher è troppo rapido per catturare gli stati logici nelle uscite del 74LS139 durante i cicli sia di lettura che di scrittura. Quello che intendiamo sottolineare è che il bit catcher "cattura" lo stato logico delle uscite del 74LS139 *prima* che queste possano avere la possibilità di riflettere propriamente lo stato degli ingressi. Per essere più chiari, elenchiamo tutti gli eventi che si sarebbero verificati se voi aveste misurato lo stato logico delle uscite del 74LS139 con il "bit catcher". La Figura 3-34 è un diagramma di temporizzazione che mostra le correlazioni di questi eventi per un ciclo di scrittura di I/O.

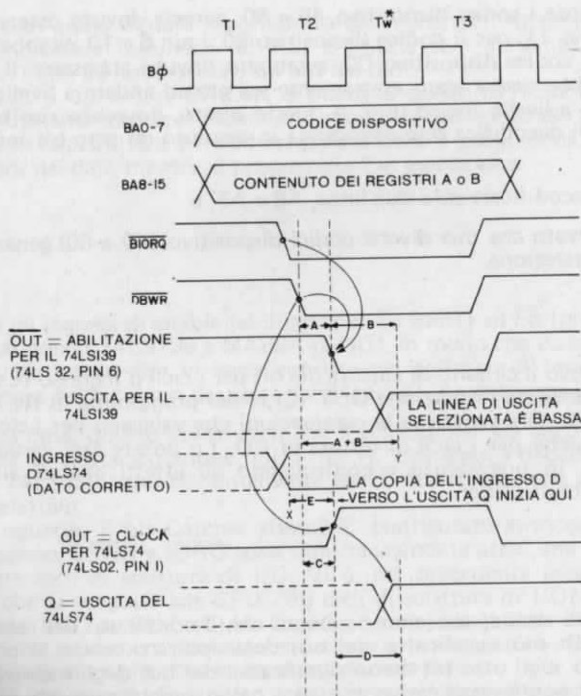


Figura 3-34. Diagramma dei tempi per la cattura dei bit di uscita dal decodificatore.

1) Viene eseguita dalla CPU l'istruzione IN (C),H o OUT (C),H. Si consideri l'istruzione OUT (C),H; ne consegue questa sequenza di eventi:
   a. Il codice dispositivo è posto su BA0-BA7. Il contenuto del registro B è posto su BA8-BA15.
   b. Sono attivati i segnali $\overline{BIORQ}$ e $\overline{DBWR}$.

2) I segnali $\overline{BIORQ}$ e $\overline{DBWR}$ sono inviati in ingresso alle due porte:

   74LS02 : NOR per i circuiti di cattura dei bit
   74LS32 : OR per i circuiti di decodifica

   Ciascuna di queste porte genererà un segnale di uscita per attuare il clock o abilitare un flip-flop D oppure un decodificatore/demultiplexer. La generazione di queste uscite necessita di *tempo*. Questo periodo di tempo è chiamato *ritardo di propagazione* (propagation delay) ed è stato indicato come A per la porta OR 74LS32 e C per la porta NOR 74LS02. Nel Data Book della SGS-ATES sono dati, per ciascuna componente, i ritardi di propagazione minima (MIN), massimo (MAX) (vedi Tavole 3-4 e 3-5). Tipico, spesso, il ritardo è diverso se la transizione è da basso ad alto o da alto a basso. Per il 74LS02, l'uscita è normalmente bassa, e la transizione che ci interessa è da basso ad alto. Dal Data Book della SGS-ATES troviamo che:

   $$3\text{ ns} < C < 10\text{ ns e tipico} = 5\text{ ns}$$

   Per il 74LS32, l'uscita è normalmente alta e la transizione che ci interessa è da alto a basso:

   $$3\text{ ns} < A < 11\text{ ns e tipico} = 7\text{ ns}$$

   Notate che vi è già una potenziale differenza durante la velocità di propagazione dei segnali attraverso il decodificatore ed i circuiti di cattura dei bit.

3) L'uscita del 74LS32 abilita il decodificatore/demultiplexer "b" del 74LS139. A causa del tempo di propagazione, vi è un ritardo alle porte $\overline{O0b}$, $\overline{O1b}$, $\overline{O2b}$ e $\overline{O3b}$ che mostrano il contenuto degli ingressi selezionati A0b ed A1b.
   Si nota che il bus degli indirizzi presenta il codice dispositivo stabile ben prima che l'impulso di abilitazione raggiunga il chip 74LS139. Il ritardo di propagazione è indicato con B nella Figura 3-34.
   Ancora la transizione da alto a basso è di preminente interesse. Dal Data Book della SGS-ATES:

   $$B < 24\text{ ns e tipico} = 17\text{ ns (non è dato nessun MIN)}$$

   Quindi, A+B è il ritardo in nanosecondi della attivazione di $\overline{BIORQ}$ e $\overline{DBWR}$ nella stabilizzazione delle uscite del 74LS139

   $$A + B < 35\text{ ns e tipico} = 24\text{ ns}$$

4) L'uscita del 74LS02 attua il clock del flip-flop 74LS74. L'ingresso D (pin 2) è riportato sulle uscite Q con un ritardo indicato con D nella Figura 3-34. Questo ritardo di fatto non è critico. Ciò che è critico è la stabilità o meno delle uscite del decodificatore/demultiplexer, che devono riflettere lo stato degli ingressi selezionati prima che il 74LS74 realizzi la copia dell'ingresso D sull'uscita Q.
   Sfortunatamente non sono necessariamente stabili.
   Il flip-flop D inizia a riportare il corrente ingresso D sull'uscita Q circa nello stesso istante in cui il decodificatore/demultiplexer è sottoposto a strobe (entro 0-8 ns). Dalla Figura 3-34 è facile vedere che la copia inizia ben prima del momento in cui l'uscita selezionata del 74LS139 sia bassa.

Quindi abbiamo individuato un'altra limitazione nell'uso del circuito di cattura dei

bit. Come tutti gli strumenti semplici e poco costosi, ha i suoi pregi, ma non è certo sempre utilizzabile. Quindi la cosa più importante che ci preme sottolineare a questo punto è che i problemi di temporizzazione possono essere critici specialmente quando i segnali si propagano sequenzialmente attraverso una serie di dispositivi.
Come potete ben immaginare, un'analisi delle temporizzazioni per circuiti complessi può risultare molto gravosa. Immaginate i requisiti di temporizzazione per il circuito nel nostro chip CPU Z80! Vi è stato un pesante utilizzo di calcolatori per verificare le temporizzazioni per questi circuiti durante la fase di progettazione.
Verificate sperimentalmente il vostro "bit catcher" con riferimento alle uscite del 74LS139 sia per le operazioni di lettura di I/O che di scrittura. Avete dei comportamenti non corretti? Se no, consideratevi fortunati.

Lasciate collegato il circuito numero 5 per un successivo uso nell'Esperimento N. 3.

ESPERIMENTO N. 3

Scopo

Lo scopo di questo esperimento è illustrare uno degli usi degli impulsi di selezione dispositivo generati sotto il controllo del software, ovvero l'invio di impulsi di strobe ad un contatore a decade. Per generare impulsi multipli di selezione dispositivo, viene usata anche l'istruzione OTIR: tali impulsi vengono contati dal contatore a decade 74LS90.

Schema del circuito (Figura 3-35)

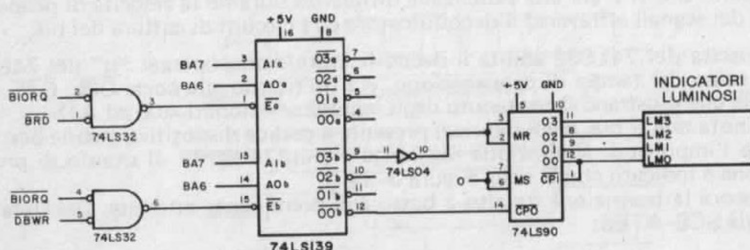


Figura 3-35. Schema N. 6.

Passo 1

Montate il circuito 74LS90 mostrato in Figura 3-35. Se non avete ancora montato il circuito di decodifica dell'Esperimento n. 2, fatelo ora; collegate il pin 12 del decodificatore 74LS139 al pin 14 del contatore 74LS90; e collegate il pin 9 del 74LS139 per mezzo di un invertitore al pin 3 del 74LS90.

Passo 2

Caricate il programma del Nanocomputer ed eseguitelo. Che cosa osservate?

Abbiamo osservato che il contatore ha registrato otto, che è uguale al numero di byte posti in uscita dell'istruzione OTIR, e quindi al numero di impulsi di selezione dispositivo generati.

Configurazione dei pin del circuito integrato (Tavola 3-6)

Tavola 3-6. Caratteristiche del T54LS90/T74LS90 – T54LS92/T74LS92 – T54LS93/T74LS93.

DESCRIPTION – The T54LS90/T74LS90, T54LS92/T74LS92 and T54LS93/T74LS93 are high-speed 4-bit ripple type counters partitioned into two sections. Each counter has a divide-by-two section and either a divide-by-five (LS90), divide-by-six (LS92) or divide-by-eight (LS93) section which are triggered by a HIGH-to-LOW transition on the clock inputs. Each section can be used separately or tied together (Q to $\overline{CP}$) to form BCD, bi-quinary, modulo-12, or modulo-16 counters. All of the counters have a 2-input gated Master Reset (Clear), and the LS90 also has a 2-input gated Master Set (Preset 9).

- **LOW POWER CONSUMPTION . . . TYPICALLY 45 mW**
- **HIGH COUNT RATES . . . TYPICALLY 50 MHz**
- **CHOICE OF COUNTING MODES . . . BCD, BI-QUINARY, DIVIDE-BY-TWELVE, BINARY**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**

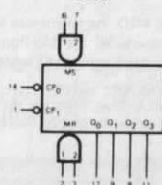
PIN NAMES

		LOADING (Note a)	
		HIGH	LOW
$\overline{CP}_0$	Clock (Active LOW going edge) Input to ÷2 Section	3.0 U.L.	1.5 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷5 Section (LS90), ÷6 Section (LS92)	2.0 U.L.	2.0 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷8 Section (LS93)	1.0 U.L.	1.0 U.L.
MR_1, MR_2	Master Reset (Clear) Inputs	0.5 U.L.	0.25 U.L.
MS_1, MS_2	Master Set (Preset-9, LS90) Inputs	0.5 U.L.	0.25 U.L.
Q_0	Output from ÷2 Section (Notes b & c)	10 U.L.	5(2.5) U.L.
Q_1, Q_2, Q_3	Outputs from ÷5 (LS90), ÷6 (LS92), ÷8 (LS93) Sections (Note b)	10 U.L.	5(2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.
c. The Q_0 Outputs are guaranteed to drive the full fan-out plus the $\overline{CP}_1$ input of the device.

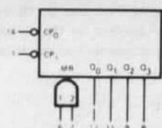
LOGIC SYMBOL

LS90



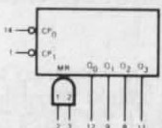
V_{CC} = Pin 5
GND = Pin 10
NC = Pins 4, 13

LS92



V_{CC} = Pin 5
GND = Pin 10
NC = Pins 2, 3, 4, 13

LS93



V_{CC} = Pin 5
GND = Pin 10
NC = Pins 4, 6, 7, 13

Tavola 3-6. Caratteristiche del T54LS90/T74LS90 – T54LS92/T74LS92 – T54LS93/T74LS93 (seguito).

FUNCTIONAL DESCRIPTION – The LS90, LS92, and LS93 are 4-bit ripple type Decade, Divide-By-Twelve, and Binary Counters respectively. Each device consists of four master/slave flip-flops which are internally connected to provide a divide-by-two section and a divide-by-five (LS90), divide-by-six (LS92), or divide-by-eight (LS93) section. Each section has a separate clock input which initiates state changes of the counter on the HIGH-to-LOW clock transition. State changes of the Q outputs do not occur simultaneously because of internal ripple delays. Therefore, decoded output signals are subject to decoding spikes and should not be used for clocks or strobes. The Q_0 output of each device is designed and specified to drive the rated fan-out plus the $\overline{CP}_1$ input of the device.

A gated AND asynchronous Master Reset ($MR_1 \cdot MR_2$) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set ($MS_1 \cdot MS_2$) is provided on the LS90 which overrides the clocks and the MR inputs and sets the outputs to nine (HLLH).

Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.

LS90

A. BCD Decade (8421) Counter – The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and a BCD count sequence is produced.

B. Symmetrical Bi-quinary Divide-By-Ten Counter – The Q_3 output must be externally connected to the $\overline{CP}_0$ input. The input count is then applied to the $\overline{CP}_1$ input and a divide-by-ten square wave is obtained at output Q_0.

C. Divide-By-Two and Divide-By-Five Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function ($\overline{CP}_0$ as the input and Q_0 as the output). The $\overline{CP}_1$ input is used to obtain binary divide-by-five operation at the Q_3 output.

LS92

A. Modulo 12, Divide-By-Twelve Counter – The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and Q_3 produces a symmetrical divide-by-twelve square wave output.

B. Divide-By-Two and Divide-By-Six Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function. The $\overline{CP}_1$ input is used to obtain divide-by-three operation at the Q_1 and Q_2 outputs and divide-by-six operation at the Q_3 output.

LS93

A. 4-Bit Ripple Counter – The output Q_0 must be externally connected to input $\overline{CP}_1$. The input count pulses are applied to input $\overline{CP}_0$. Simultaneous divisions of 2, 4, 8, and 16 are performed at the Q_0, Q_1, Q_2 and Q_3 outputs as shown in the truth table.

B. 3-Bit Ripple Counter – The input count pulses are applied to input $\overline{CP}_1$. Simultaneous frequency divisions of 2, 4, and 8 are available at the Q_1, Q_2, and Q_3 outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple-through counter.

Programma PULSR

Codice oggetto	Codice sorgente	Commenti
0E 20	PULSR: LD C,020H	; Carica il registro C con il codice dispositivo
21 00	LD HL,15H	; Carica la coppia di registri HL con l'indirizzo di ; memoria iniziale
06 08	LD B,08H	; Carica il registro B con il contatore del byte
D3 C0	OUT (0C0H),A	; Azzera il contatore a decade
ED B3	OTIR	; Poni in uscita la stringa di byte che inizia all'indi- ; rizzo HL, di lunghezza (B), sulla porta (C)
76	HALT	; Arresta il Nanocomputer

Passo 3

Cambiate l'istruzione LD B,08H per inizializzare il registro B in ognuno dei seguenti conteggi dei byte in uscita; eseguite il programma per ognuno dei nuovi conteggi e registrate il conteggio sul 74LS90 leggendo gli indicatori luminosi.

Contenuto nel registro B	Valore del contatore (rappresent. decimale)
01	
02	
03	
04	
05	
06	
07	
08	
09	
0A	
0B	
0C	
0D	
0E	
0F	
10	

Abbiamo osservato che, per i valori che vanno da 01 a 09, il contatore ha letto da 1 a 9. La prima sorpresa è stata vedere il contenuto del registro B uguale a 0A; il contatore ha letto 0 invece di A. Per 0B, il contatore ha letto 1; per 0C il contatore ha letto 2; e, il contatore ha proseguito aumentando sempre di uno finchè il contenuto del registro B è arrivato a 14, momento in cui il contatore è ritornato a zero. Perciò, il contatore effettua cicli da 0 a 9, e poi di nuovo a 0. Ecco perchè il 74LS90 viene chiamato contatore a *decade* . . . esso effettua i suoi cicli attraverso 10 letture distinte. Un'altra definizione che viene data di questo contatore è "contatore *divisore per dieci*", perchè mostra sempre il resto risultante della divisione per dieci del numero di impulsi in esso inseriti. Qual'è la relazione esistente fra il contenuto B e la lettura del contatore. Il registro B specifica il numero di cicli di uscita che devono essere eseguiti dall'istruzione OTIR. Dato che ogni ciclo di uscita genera un impulso di selezione dispositivo sul pin di uscita $\overline{O0b}$ del decodificatore "b" del chip 74LS139, e questo pin è collegato all'ingresso del contatore ($\overline{CP0}$, pin 14) del 74LS90, ogni ciclo di uscita viene realizzato come conteggio incrementale sull'indicatore luminoso. Verificate che il codice dispositivo 20 corrisponde all'uscita $\overline{O0b}$ del decodificatore/demultiplexer.

Passo 4

Che scopo ha il collegamento fra il pin 9 del decodificatore e il pin 2 del contatore?

Quando viene eseguita l'istruzione OUT (0C0H),A, viene generato un impulso di selezione dispositivo sul pin 9, l'uscita $\overline{O3b}$ del decodificatore "b" che, dopo essere passata attraverso un invertitore, fornisce un impulso positivo di strobe al contatore di impulsi, sull'ingresso di azzeramento, il pin 3. Perciò il filo che collega decodificatore e contatore, unitamente al software, azzera il contatore subito prima che esso venga usato per contare gli impulsi di selezione generati dall'istruzione OTIR.

Passo 5

Variando il codice dispositivo caricato nel registro C e cambiando i collegamenti in modo che il nuovo indirizzo di selezione dispositivo risultante venga trasmesso all'ingresso del contatore del 74LS90, è possibile verificare la corrispondenza fra il contenuto del registro C e l'impulso di selezione dispositivo generato del decodificatore/demultiplexer 74LS139.

Passo 6

Tentate un esperimento analogo a quest'ultimo con l'istruzione INDR e il decodificatore "a" del 74LS139. Naturalmente, esperimenti analoghi possono essere effettuati anche con le istruzioni OTDR e INIR.

Passo 7

Sostituite l'istruzione HALT alla locazione 013B del suddetto programma con la istruzione di restart (ripristino) RST 38H, codice esadecimale FF. Eseguite il programma così variato con byte diversi caricati nel registro B, cioè variate il byte nella istruzione LD B,08H. Che cosa osservate?

Abbiamo osservato che l'uscita del contatore a decade 74LS90 aveva sempre come valore 0.
Perchè una modifica così semplice crea un risultato così disastroso? La ragione sta nel fatto che, per mezzo dell'istruzione RST 38H, abbiamo rimandato il controllo del Nanocomputer al sistema operativo, un programma che voi *non* avete scritto. In qualunque momento affidate il controllo ad un software con il quale non avete molta familiarità, dovete essere preparati ad ottenere risultati imprevedibili. In questo caso, il sistema operativo del Nanocomputer, non appena si trova a gestire nuovamente il controllo, inizia ad aggiornare costantemente il display ed a controllare la tastiera stessa per verificare se vi sono tasti premuti.
Questi compiti coinvolgono un dispositivo di I/O, cioè l'esecuzione ripetuta delle istruzioni IN e OUT. Dato che il nostro semplice circuito non decodifica *in assoluto* i codici dispositivo, esso ha una tendenza a venire attivato più spesso di quanto dovrebbe. In particolare l'ingresso di *clear*, che viene attivato ogni volta che BA7 e BA6 sono entrambi a livello logico zero, viene attivato con ogni istruzione IN o OUT eseguita dal sistema operativo del Nanocomputer mentre svolge i suoi compiti di I/O della tastiera e del display.

Il risultato di tutto questo è che, mentre l'esecuzione dell'istruzione OTIR può portare il contatore a registrare un conteggio di non zero, questo dura solo per pochi millisecondi, perchè il sistema operativo, per mezzo dell'istruzione RST 38H, riprende presto il controllo ed azzera il contatore. E' quest'ultimo effetto che noi vediamo.

Vi abbiamo descritto questo fenomeno allo scopo di fornirvi una prima esperienza pratica necessaria per affrontare i fastidiosi inconvenienti che si verificano quando si ha a che fare con un software con cui non si ha dimestichezza. Chiaramente, una soluzione è quella di familiarizzare appieno con tutto il software che vi capiterà di usare. Sfortunatamente, questo non è quasi mai possibile. Voi ed il sistema operativo del vostro Nanocomputer costituisce un primo esempio. Quindi . . . siete stati avvisati!

ESPERIMENTO N. 4

Scopo

Lo scopo di questo esperimento è quello di far vedere come si usa il dispositivo porta NAND a 8 ingressi 74LS30, per generare gli impulsi di selezione memoria per accedere a 256 byte di memoria statica sia in lettura sia in scrittura, per una memoria ad accesso casuale (R/W RAM).

Per questo esperimento prendiamo in considerazione due dispositivi di memoria statica del tipo Intel 2101A da 256 x 4 bit. La realizzazione pratica di questo esperimento richiede il cablaggio di un circuito abbastanza complesso, ma pensiamo sia una fatica che ne valga le pena. Ad ogni modo è permesso un tempo sufficiente per il montaggio e collaudo completo del circuito.

Schema del circuito (Figura 3-36)

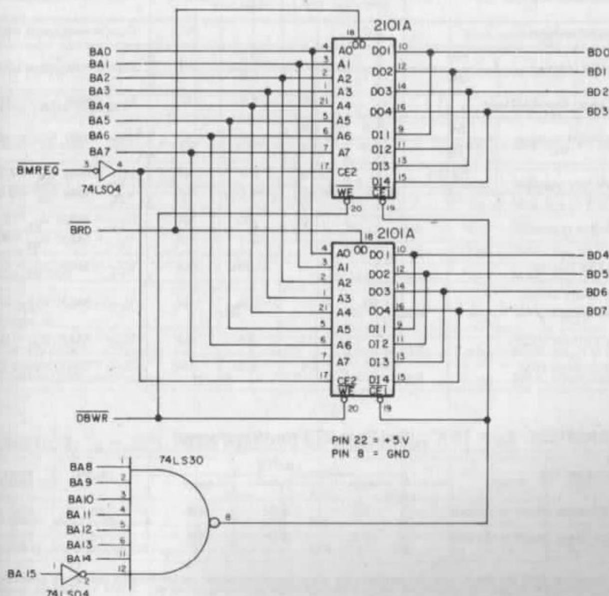
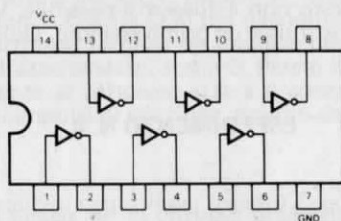


Figura 3-36. Schema N. 7.

Configurazioni dei pin dei circuiti integrati (Tavole 3-7, 3-8, 3-9)

Tavola 3-7. Caratteristiche del T54LS04/T74LS04.

HEX INVERTER



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS04X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS04X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type: D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.2	2.4	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			3.6	6.6	mA	V_{CC} = MAX, Inputs Open

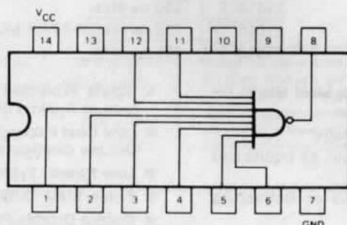
AC CHARACTERISTICS: T_A = 25°C **(See Page 273 for Waveforms)**

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25 C.
3. Not more than one output should be shorted at a time.

Tavola 3-8. Caratteristiche del T54LS30/T74LS30.

8-INPUT NAND GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS30X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS30X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			0.35	0.5	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			0.6	1.1	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: $T_A = 25°C$ (See Page 273 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output		6.5	12	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output		12.5	15	ns	C_L = 15 pF

NOTES
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tavola 3-9. Caratteristiche della RAM Statica Intel 2101A/8101A-4* 256x4 bit.

2101A-2	250 ns Max.
2101A	350 ns Max.
2101A-4	450 ns Max.

- **256 x 4 Organization to Meet Needs for Small System Memories**
- **Single +5V Supply Voltage**
- **Directly TTL Compatible: All Inputs and Output**
- **Statis MOS: No Clocks or Refreshing Required**
- **Simple Memory Expansion: Chip Enable Input**
- **Inputs Protected: All Inputs Have Protection Against Static Charge**
- **Low Cost Packaging: 22 Pin Plastic Dual In-Line Configuration**
- **Low Power: Typically 150 mW**
- **Three-State Output: OR-Tie Capability**
- **Output Disable Provided for Ease of Use in Common Data Bus Systems**

The Intel® 2101A is a 256 word by 4-bit static random access memory element using N-channel MOS devices integrated on a monolithic array. It uses fully DC stable (static) circuitry and therefore requires no clocks or refreshing to operate. The data is read out nondestructively and has the same polarity as the input data.

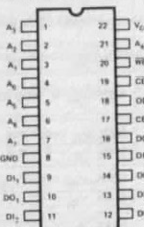
The 2101A is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. Two chip-enables allow easy selection of an individual package when outputs are OR-tied. An output disable is provided so that data inputs and outputs can be tied for common I/O systems. The output disable function eliminates the need for bi-directional logic in a common I/O system.

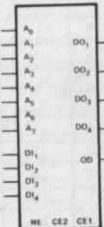
The Intel® 2101A is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance, easy-to-use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

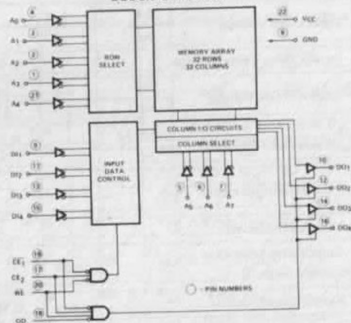
Intel's silicon gate technology also provides excellent protection against contamination. This permits the use of low cost plastic packaging.

PIN CONFIGURATION



LOGIC SYMBOL





PIN NAMES

DI_1-DI_4	DATA INPUT	CE_2	CHIP ENABLE 2
A_0-A_7	ADDRESS INPUTS	OD	OUTPUT DISABLE
$\overline{WE}$	WRITE ENABLE	DO_1-DO_4	DATA OUTPUT
$\overline{CE}_1$	CHIP ENABLE 1	V_{CC}	POWER (+5V)

*All 8101A-4 specs are identical to the 2101A-4 specs.

Tavola 3-9. Caratteristiche della RAM Statica 2101A/8101A-4 (seguito).

2101A FAMILY

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . . -10°C to 80°C
Storage Temperature . . . . -65°C to +150°C
Voltage On Any Pin
With Respect to Ground . . . . -0.5V to +7V
Power Dissipation . . . . 1 Watt

COMMENT:

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

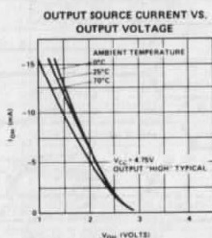
D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified.

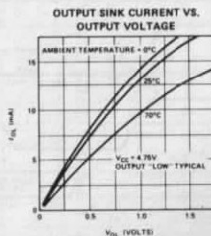
Symbol	Parameter		Min.	Typ.[1]	Max.	Unit	Test Conditions
I_{LI}	Input Current			1	10	μA	V_{IN} = 0 to 5.25V
I_{LOH}	Data Output Leakage Current			1	10	μA	Output Disabled, V_{OUT}=4.0V
I_{LOL}	Data Output Leakage Current			-1	-10	μA	Output Disabled, V_{OUT}=0.45V
I_{CC1}	Power Supply Current	2101A, 2101A-4		35	55	mA	V_{IN} = 5.25V, I_O = 0mA
		2101A-2		45	65		T_A = 25°C
I_{CC2}	Power Supply Current	2101A, 2101A-4			60	mA	V_{IN} = 5.25V, I_O = 0mA
		2101A-2			70		T_A = 0°C
V_{IL}	Input "Low" Voltage		-0.5		+0.8	V	
V_{IH}	Input "High" Voltage		2.0		V_{CC}	V	
V_{OL}	Output "Low" Voltage				+0.45	V	I_{OL} = 2.0mA
V_{OH}	Output "High" Voltage	2101A, 2101A-2	2.4			V	I_{OH} = -200μA
		2101A-4	2.4			V	I_{OH} = -150μA

TYPICAL D.C. CHARACTERISTICS

OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



NOTES: 1. Typical values are for T_A = 25°C and nominal supply voltage.

Tavola 3-9. Caratteristiche della RAM Statica 2101A/8101A-4 (seguito).

A.C. CHARACTERISTICS FOR 2101A-2 (250 ns ACCESS TIME)

READ CYCLE $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	250			ns	$t_r, t_f = 20ns$ Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_A	Access Time			250	ns	
t_{CO}	Chip Enable To Output			180	ns	
t_{OD}	Output Disable To Output			130	ns	
t_{DF} [3]	Data Output to High Z State	0		180	ns	
t_{OH}	Previous Read Data Valid after change of Address	40			ns	

WRITE CYCLE

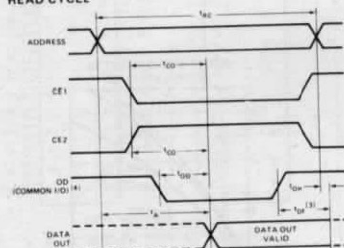
Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	170			ns	$t_r, t_f = 20ns$ Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_{AW}	Write Delay	20			ns	
t_{CW}	Chip Enable To Write	150			ns	
t_{DW}	Data Setup	150			ns	
t_{DH}	Data Hold	0			ns	
t_{WP}	Write Pulse	150			ns	
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

CAPACITANCE [2] $T_A = 25°C$, f = 1MHz

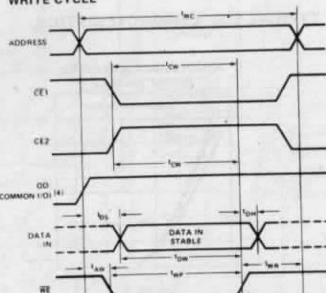
Symbol	Test	Limits (pF) Typ.[1]	Max.
C_{IN}	Input Capacitance (All Input Pins) V_{IN} = 0V	4	8
C_{OUT}	Output Capacitance V_{OUT} = 0V	8	12

WAVEFORMS

READ CYCLE



WRITE CYCLE



NOTES
1. Typical values are for $T_A = 25°C$ and nominal supply voltage.
2. This parameter is periodically sampled and is not 100% tested.
3. t_{DF} is with respect to the trailing edge of $\overline{CE_1}$, CE_2, or OD, whichever occurs first.
4. OD should be tied low for separate I/O operation.

Tavola 3-9. Caratteristiche della RAM Statica 2101A/8101A-4 (seguito).

2101A (350 ns ACCESS TIME)
A.C. CHARACTERISTICS

READ CYCLE T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	350			ns	t_r, t_f = 20ns Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_A	Access Time			350	ns	
t_{CO}	Chip Enable To Output			240	ns	
t_{OD}	Output Disable To Output			180	ns	
t_{DF} [2]	Data Output to High Z State	0		150	ns	
t_{OH}	Previous Read Data Valid after change of Address	40			ns	

WRITE CYCLE

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	220			ns	t_r, t_f = 20ns Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_{AW}	Write Delay	20			ns	
t_{CW}	Chip Enable To Write	200			ns	
t_{DW}	Data Setup	200			ns	
t_{DH}	Data Hold	0			ns	
t_{WP}	Write Pulse	200			ns	
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

2101A-4 (450 ns ACCESS TIME)
A.C. CHARACTERISTICS

READ CYCLE T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	450			ns	t_r, t_f = 20ns Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_A	Access Time			450	ns	
t_{CO}	Chip Enable To Output			310	ns	
t_{OD}	Output Disable To Output			250	ns	
t_{DF} [2]	Data Output to High Z State	0		200	ns	
t_{OH}	Previous Read Data Valid after change of Address	40			ns	

WRITE CYCLE

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	270			ns	t_r, t_f = 20ns Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_{AW}	Write Delay	20			ns	
t_{CW}	Chip Enable To Write	250			ns	
t_{DW}	Data Setup	250			ns	
t_{DH}	Data Hold	0			ns	
t_{WP}	Write Pulse	250			ns	
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

NOTES: 1. Typical values are for T_A = 25°C and nominal supply voltage.
2. t_{DF} is with respect to the trailing edge of $\overline{CE_1}$, CE_2, or OD, whichever occurs first.

Programma MEM1

Codice oggetto	Codice sorgente			Commenti
3EFF	MEM1:	LD	A,0FFH	; Inizializza l'accumulatore
3C	LOOP4:	INC	A	; Prepara test di memoria per il ; valore seguente
32007F		LD	(7F00H),A	; Inizializza locaz. 7F00 con acc.
01FF00		LD	BC,0FFH	; BC = contatore di byte per LDIR
11017F		LD	DE,7F01H	; DE = puntatore per memoria de- ; stinazione
21007F		LD	HL,7F00H	; Puntatore per memoria di prelievo
EDB0		LDIR		; Carica 7F00–7FFF con l'accumu- ; latore
010001	CHECK	LD	BC,0100H	; Verifica caricamento precedente
2100F		LD	HL,7F00H	; Puntatore locazione da verificare ; contatore di byte
EDA1	NXTLOC	CPI		; Paragona (HL) con A
200B		JR	NZ,ERROR	; Se diversi indica errore
E23E01		JP	PO,NEXXT	; Parity = 0 indica che BC = 0000, ; testa byte prossimo
18F7		JR	NXTLOC	; Se BC ≠ 0 va alla locazione se- ; guente
FEFF	NEXXT	CP	0FFH	; Test se A = FF
20DE		JR	NZ,LOOP4	; Se no, test byte prossimo
1820		JR	END	; Se si test finito
08	ERROR	EX	AF,AF'	; Display byte errato usando due ; routine del sistema operativo del ; Nanocomputer
3E70		LD	A,70H	
08		EX	AF,AF'	
3EE0		LD	A,0E0H	
32E50F		LD	(ADDH),A	; Carica 'E'in digit sinistro display
2B		DEC	HL	; HL = puntatore locaz. errata
7D		LD	A,L	
32E20F		LD	(DATAL),A	
7C		LD	A,H	
32E30F		LD	(DATAH),A	
21B90F		LD	HL,LEDL	
11E50F		LD	DE,ADDH	
CD7CFA		CALL	CONVDI	
CD09F9	ERRLP	CALL	DISPL	
18FB		JR	ERRLP	
08	END	EX	AF,AF'	; Display tutte 'F' se test buono
3E00		LD	A,00H	
08		EX	AF,AF'	
3EFF		LD	A,0FFH	
32E50F		LD	(ADDH),A	
32E40F		LD	(ADDL),A	
32E30F		LD	(DATAH),A	
32E20F		LD	(DATAL),A	
21B90F		LD	HL,LEDL	
11E50F		LD	DE,ADDH	
CD7CFA		CALL	CONVDI	
CD09F9	OK	CALL	DISPL	
19FB		JR	OK	

Passo 1

In questo esperimento dovrete manipolare dei dispositivi MOS chiamati 2101A R/W RAM. Per assicurarsi che siano ridotte al minimo le probabilità di danneggiare questi circuiti integrati, si prega di leggere e seguire attentamente le norme suggerite nella Appendice relativa alla manipolazione dei dispositivi MOS.

Mediante il circuito precedente, voi potete aggiungere 256 byte di R/W Ram al vostro Nanocomputer. Studiate con attenzione la descrizione del dispositivo 2101A tratta dal Data Book Intel. Dovreste accertarvi di aver capito alla perfezione le seguenti caratteristiche del dispositivo 2101A Ram a quelle del circuito precedente.

a. Il chip contiene 256 parole da 4 bit. Ci sono otto piedini di ingresso per gli indirizzi, i quali servono per determinare quale delle 256 (2^8) parole da 4 bit viene indirizzata. I due dispositivi 2101A funzionano insieme, per cui con un indirizzo di otto bit si ottiene un byte di otto bit, quattro per ogni dispositivo.

b. Sul chip 2101A ci sono due gruppi di quattro piedini cada uno che servono per lo scambio dei dati. Un gruppo di quattro linee serve solamente per i dati in Ingresso, mentre il secondo serve per i dati in Uscita. I dati di ingresso sono dati che vengono scritti nella memoria dalla CPU Z80 durante un ciclo di scrittura in memoria.

c. Attivazione e disattivazione dei due gruppi da quattro piedini. DIn e DOn (dove n = 1, 2, 3, 4), sono controllati da quattro segnali di controllo: CE1, $\overline{CE2}$, $\overline{WE}$, OD. $\overline{CE}$ sta per abilitazione dispositivo (chip enable), WE sta per abilitazione scrittura (write enable), mentre OD significa uscite non abilitate (output disable).
Dal diagramma a blocchi al fondo della prima pagine della specifica Intel 2101A, è evidente che:

(1) I buffer di ingresso per i dati in Input sono attivati se e solo se $\overline{CE1}$, $\overline{CE2}$ e $\overline{WE}$ sono tutti a livello attivo. Nello schema del circuito di memoria per questo esperimento, $\overline{CE1}$ è collegato all'uscita della porta NAND a 8 ingressi 74LS30, $\overline{CE2}$ è collegato all'invertitore del segnale $\overline{BMREQ}$, (attivo basso), e $\overline{WE}$ è collegato al segnale $\overline{DBWR}$.
Questi tre segnali sono attivi se e solo se viene eseguito un ciclo di scrittura in memoria, e la parte alta del bus degli indirizzi contiene 7F hex.

(2) I buffer di uscita sono attivati soltanto se $\overline{CE1}$ e $\overline{CE2}$ sono attivi e $\overline{WE}$ e OD sono a livello logico inattivo.
In conseguenza del fatto che OD è collegato a BRD, OD non è attivo solo se $\overline{BRD}$ è attivo. Ne consegue che i buffer per i dati in uscita sono abilitati soltanto se si sta eseguendo un ciclo di lettura in memoria (quindi $\overline{WR}$ e $\overline{DBWR}$ non sono attivi) e la parte alta del bus degli indirizzi contiene 7F hex.

Passo 2

Cablare il circuito mostrato precedentemente. A causa della complessità del circuito suddetto, vi elenchiamo le seguenti indicazioni per aiutarvi a minimizzare il numero dei collegamenti che dovete realizzare:

a. Disposizione dei dispositivi nella basetta per esperimenti: tutti i bus di indirizzi e di dati devono essere collegati tra la RAM sulla basetta e gli zoccoli B e C. Il modo migliore per sistemare i dispositivi 2101A e 74LS30 e il più intorno possibile ai due zoccoli. I due 2101A dovrebbero essere affiancati tra loro, con 74LS40 alla loro destra o sinistra. La posizione del 74LS04 non è critica.

b. Collegare per i primi piedini dell'alimentazione. Notate che ci sono parecchi segnali che vanno agli stessi piedini su entrambi i dispositivi 2101A, per esempio, la parte bassa del bus degli indirizzi (BA0-BA7), $\overline{BMREQ}$, $\overline{BRD}$ e $\overline{DBWR}$. Collegare insieme ognuna di queste coppie con fili corti rossi e neri.

c. Notare che i piedini dei dati in ingresso (DI1 - DI14) sui due chip 2101A sono collegati ai loro corrispondenti piedini dei dati in uscita (DO1 - DO4). Collegare adesso queste coppie, usando ancora dei fili corti.

d. Collegare i bus degli indirizzi, il bus dei dati ed i segnali di controllo degli zoccoli B e C ai rispettivi piedini dei dispositivi 2101A, 74LS30 e 74LS04. E' bene utilizzare fili di diverso colore alternati per i bus dei dati e degli indirizzi.

Passo 3

Il prossimo passo è quello di cercare di leggere e scrivere dati da e nella memoria che avete appena aggiunto al vostro Nanocomputer. In ogni caso prima di fare questo, si devono fare le considerazioni seguenti:

a. *A quale zona di indirizzi è assegnata la memoria?*
Per rispondere a questa domanda dovete guardare lo schema del circuito. La chiave del problema è come sono usati i segnali del bus degli indirizzi. Notare che la parte bassa è collegata direttamente agli ingressi dei dispositivi 2101A. Queste linee portano gli otto bit meno significativi degli indirizzi di tutte le locazioni indirizzabili dalla CPU Z80. Il campo dei valori possibili per questi 8 bit è da 00 a FF, cioè 256 configurazioni diverse.
Cosa si può dire per gli otto bit più significativi delle linee di indirizzamento?, esse sono collegate agli ingressi del dispositivo 74LS30. L'uscita di questa porta NAND è attiva (livello basso) se e solo se:

BA –	15	14	13	12	11	10	9	8
	0	1	1	1	1	1	1	1

In altre parole, l'uscita del 74LS30, il quale insieme al segnale $\overline{BMREQ}$ serve come abilitazione dispositivo, è attiva soltanto se l'indirizzo è del tipo 7FXX, dove XX può essere un valore qualunque.
Quindi, la risposta alla domanda è che la memoria 2101A è selezionata dagli indirizzi compresi nel campo da 7F00 e 7FFF.

b. *Come si accede alla memoria 2101A?*
Questo è facile. Semplicemente basta usare il sistema operativo del Nanocomputer. La memoria può essere letta commutando il selettore nella posizione MEM, caricando l'indirizzo 7F00 e premendo il tasto INC.

Cercate di leggere le locazioni di memoria da 7F00 a 7F0E. Cosa osservate?

Noi osserviamo una serie dei byte differenti tra loro. Se trovate che tutte le locazioni lette sono FF è molto probabile che ci sia un errore nel vostro circuito. Provate a cambiare il contenuto di alcune locazioni in modo tale da leggere 00.
Come fate questo?

Di nuovo la risposta è, usando il sistema operativo del Nanocomputer. Impostate direttamente l'indirizzo che si vuole cambiare, premere il tasto LA, impostare il nuovo dato e premere il tasto ST. Verificate ora che il dato sia stato introdotto correttamente.
Se la locazione contiene un valore diverso da quello impostato voi avete sicuramente un errore nel circuito.

Vi diamo alcuni suggerimenti per il collaudo del circuito:

a. Isolate la parte di decodifica del circuito 74LS30 e parti ad esso associate, dal resto del circuito; verificate se questa funzioa.
Questo può essere fatto con un circuito similare al 'pulse catcher' che avete cablato per un esperimento precedente.
Per accertarsi che l'uscita del 74LS30 sia attiva solo quando sulla parte alta del bus degli indirizzi c'è 7F, cablare un circuito "pulse catcher" come mostrato nello schema N. 8 della Figura 3-37.

Figura 3-37. Schema N. 8.

Spegnete LM0, premendo momentaneamente PO. Resettare il Nanocomputer, provate ora ad accedere a varie locazioni di memoria, fuori dal campo 7EFF-7FFF.
L'indicatore luminoso dovrebbe in questo modo restare spento in quanto l'uscita della porta NAND 74LS30 non deve cambiare il suo stato logico, bensì deve restare a livello logico alto.
Cercate ora di leggere una locazione qualunque nel campo 7F00-7FFF. Dovreste osservare che l'indicatore luminoso LM0 si accende. Questo perchè il sistema operativo del Nanocomputer, quando accede ad uno qualunque di questi indirizzi, deve mettere 7F nella parte alta del bus degli indirizzi.
Se la parte di decodifica del circuito funziona come descritto precedentemente, questa è cablata correttamente.

b. Il resto del circuito può essere collaudato utilizzando procedimenti simili.

Passo 4

Caricate il programma precedente nella memoria del vostro Nanocomputer. Detto programma è un semplice test della memoria. Se il test della memoria che avete appena aggiunto al vostro Nanocomputer, da esito negativo, potete concludere che avete alcune celle di memoria non buone.
Ad ogni modo, se la vostra memoria passa il test, non dovete concludere che essa è completamente buona.
I metodi di verifica delle memorie esulano dagli scopi di questo libro. E' sufficiente dire che i sistemi per verificare il funzionamento corretto delle memorie sono stati fonte di ricerca. Alcuni test di memoria sono talmente lunghi che se si arrivasse alla loro fine, durerebbero giorni e giorni.
Il diagramma di flusso per il programma di test della memoria con inizio nella locazione MEM1 è mostrato in Figura 3-38.
In pratica il programma carica ognuno dei 256 bytes da 7F00 a 7FFF con tutte le possibili configurazioni di 8 bit e verifica se il caricamento è stato fatto correttamente. Il 'loop' esterno cambia la configurazione degli 8 bit, mentre il 'loop' interno

usa le istruzioni LDIR e CPI per effettuare il caricamento e la verifica rispettivamente. Questo sistema di test difetta proprio nella sua regolarità. Un sistema più sofisticato dovrebbe cambiare solamente *un* byte di memoria per volta e verificare quindi se sono cambiate altre locazioni di memoria.
Inoltre un sistema di test più elaborato dovrebbe effettuare delle variazioni nell'ordine di scrittura e lettura di ogni byte.
Il programma MEM1 procede nella scrittura e lettura in ordine ascendente ed in modo sequenziale.

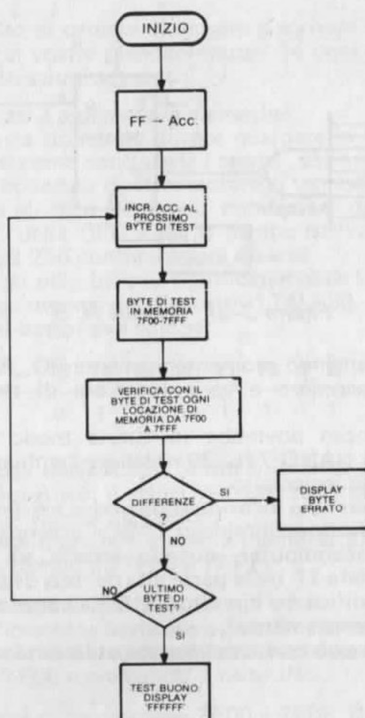


Figura 3-38. Diagramma di flusso del Test della memoria MEM1.

Vi sono due uscite possibili dal test di memoria MEM1: buono e scarto.
Se la memoria passa tutto il test, comparirà FFFFFF sul display. Se invece la memoria non pssa il programma di test, il display mostrerà 'E XXXX', dove XXXX è l'indirizzo della locazione di memoria dove il test da scarto.
Eseguite il programma di test di memoria MEM1.
Cosa osservate?

Noi osserviamo che il display resta spento per alcuni secondi dopo di che si accende con la scritta 'FFFFFF'.

Questo significa che la nostra memoria passa il test. Con un poco di buona fortuna voi osserverete lo stesso risultato, sebbene è possibile che la vostra memoria non passi il test. Se la vostra memoria da 'scarto', usate il tasto BREAK per ritornare il controllo al sistema operativo del Nanocomputer, senza cambiare il contenuto dei registri. Fate visualizzare sul display il contenuto della memoria sbagliata.

Il messaggio di errore del test della memoria è stato generato in quanto il contenuto della locazione sbagliata era diverso da quello aspettato.
Per vedere quale contenuto doveva esserci, potete vedere il contenuto della cella immediatamente precedente oppure il contenuto dell'accumulatore.
Utilizzando il sistema operativo del Nanocomputer, cercate di scrivere e leggere alcuni byte di test nella locazione di memoria sbagliata. Potete in questo caso ottenere oppure no delle differenze tra ciò che scrivete e ciò che leggete in quanto l'errore può essere dovuto ad una speciale combinazione di eventi accaduti durante l'esecuzione del programma di test MEM1.
Inoltre potrebbe essere possibile anche un *difetto intermittente*, per il quale è impossibile stabilire quando esso capita. Per esempio, ripetute esecuzioni del programma di test possono dare sia tanti test buoni che tanti test scarti.
Come abbiamo detto precedentemente, trovare celle di memoria sbagliata può essere un processo estremamente complicato.

Passo 5

Permetteteci ora di trarre in inganno il test della memoria facendo in modo che dia scarto per una locazione scelta a piacere.
Questo si può fare nel modo seguente:

a. Mettere un "breakpoint" subito dopo l'istruzione LDIR, cioè nella locazione CHECK.
b. Eseguite il programma. Quando viene trovato il "breakpoint", caricate la locazione di memoria 7F13 con un byte diverso da 00. Alterando cioè il processo di caricamento e verifica in memoria.
c. Togliete il "breakpoint".
d. Continuate l'esecuzione del programma.

Dovreste osservare il seguente messaggio di errore sul display:

E 7F13

Altri messaggi di errore sono ottenibili facilmente, seguendo la procedura precedente ma cambiando una differente locazione di memoria.

Passo 6

Vediamo ora come è possibile cambiare le locazioni dei 256 byte di RAM nei due dispositivi 2101A in una regione di memoria diversa, per esempio, in BF00-BFFF.
Come potete fare questo?
Se avete un'idea, fatelo e provatelo.

Il modo più semplice di assegnare alla RAM, nei dispositivi 2101A, la regione BF00 - BFFF è quello di cambiare le connessioni di BA14 e BA15 nella porta NAND 74LS30 e dell'invertitore 74LS04 come in Figura 3-39.

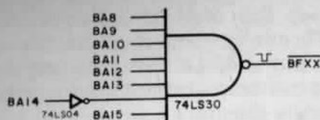


Fig. 3-39. Il dispositivo invertitore 74LS04 per assegnare la RAM alla regione BF00-BFFF.

In questo modo il dispositivo 74LS30 darà un livello attivo (basso) se e solo se si avrà una configurazione sul bus degli indirizzi pari a:

BA —	15	14	13	12	11	10	9	8
	1	0	1	1	1	1	1	1

Passo 7

Utilizzate gli invertitori restanti del dispositivo 74LS04 per assegnare alla memoria le zone:

3F00 - 3FFF
e
C000 - C0FF

Disegnate il circuito per cablare la porta NAND 74LS30 e gli invertitori 74LS04 per ognuna delle regioni di memoria suddette.
Potete inoltre verificare se lo schema per entrambe è corretto e provare il loro funzionamento.

'NON SMONTATE IL CIRCUITO DI MEMORIA, IN QUANTO E' RICHIESTO NEL PROSSIMO ESPERIMENTO'.

ESPERIMENTO N. 5

Scopo

Lo scopo di questo esperimento è quello di far vedere le differenze tra le RAM statiche e le RAM dinamiche, e l'importanza di queste differenze per quanto riguarda l'utilizzo degli stati di attesa (wait).
Questo esperimento utilizza il software e l'hardware dell'Esperimento N. 4

Configurazione dei pin dei circuiti integrati - Vedi Tavola 3-8.

Diagramma del circuito - Vedi Figura 3-36.

Programma XFER

Codice oggetto	Codice simbolico	Commenti
016600	XFER: LD BC,OK+5H-MEM1	; Prepara n° byte nel progr. MEM1 ; per LDIR = OK+5H-MEM1
11007F	LD DE,7F00H	; Memoria di destinazione = ram ; statica
211E01	LD HL,MEM1	; Memoria sorgente = quella del ; prog.
EDB0	LDIR	; Esegue trasferimento
FF	RST 38H	; Ritorno controllo a sist. operativo

Passo 1

La 256 byte x 4 2101A R/W RAM è una memoria STATICA a lettura/scrittura ad accesso casuale. Il resto R/W RAM del Nanocomputer, la cui regione è 0000-0FFF è invece una memoria *dinamica*.
Ci sono varie differenze tra le RAM *statiche* e le *dinamiche*.
Gli elementi base di immagazzinamento nelle RAM statiche sono i flip-flop. Una volta posizionati a livello 0 o a livello 1, i flip-flop mantengono il loro valore finchè sono alimentati.
Le RAM dinamiche hanno invece come organi base per l'immagazzinamento delle capacità.
La carica di queste capacità determina quando il bit di memoria è a livello logico 0 oppure 1.
Sfortunatamente una capacità con livello 1 perde la sua carica diventando a livello logico 0, e viceversa, nel giro di pochi millisecondi.
In conseguenza di ciò, le memorie dinamiche richiedono una operazione di rinfresco (refresh) ogni pochi millisecondi per ricaricare le loro capacità.
Questo fa sì che lavorare con le RAM dinamiche presenti qualche difficoltà in più rispetto alle RAM statiche. Per contro le RAM dinamiche consumano meno, hanno una maggiore densità di integrazione, e costano meno delle RAM statiche.
Quindi ogni tipo ha i suoi vantaggi e svantaggi, e sono comunque largamente usate in diverse applicazioni.

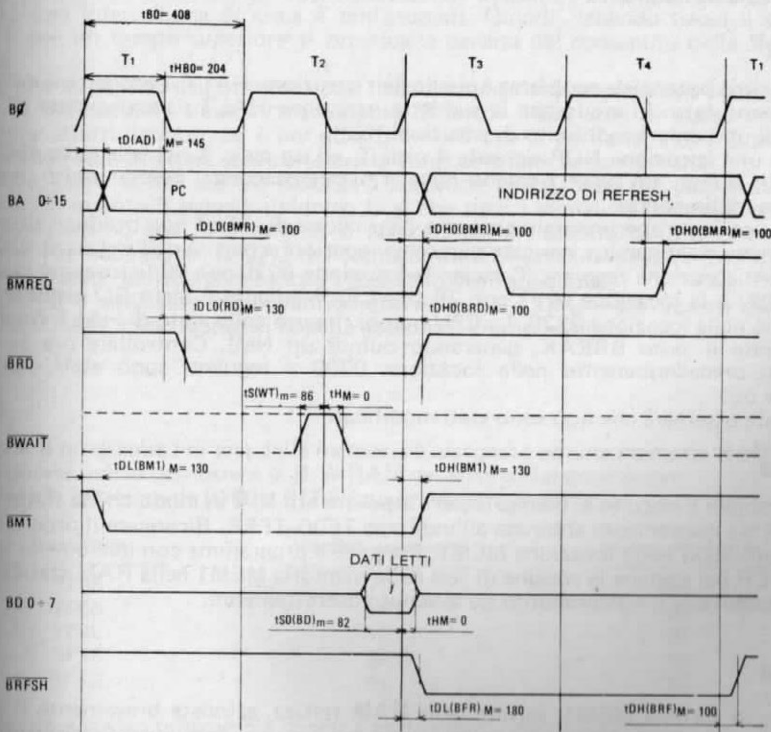


Figura 3-40. Ciclo di prelievo (fetch) in memoria con operazione di rinfresco.

Entrambe le RAM statiche e dinamiche non conservano l'informazione se viene a mancare loro la alimentazione, per questo sono chiamate memorie volatili.

Notate che la memoria EPROM che contiene il sistema operativo del Nanocomputer è *non volatile.*

Nel Capitolo 1, abbiamo detto che la CPU Z80 genera i segnali necessari per il rinfresco RAM dinamiche più diffuse.

I segnali per il rinfresco escono durante i cicli di fetch in memoria (M1), usando le linee del bus degli indirizzi, il $\overline{BMREQ}$ e $\overline{BRFSH}$.

Un ciclo di fetch in memoria è mostrato in Figura 3-40.

Siccome le R/W RAM dinamiche richiedono il rinfresco ogni qualche millisecondo, la CPU Z80 deve eseguire cicli di fetch in memoria con una frequenza tale da poter evitare la 'perdita' dell'informazione nella memoria dinamica; la maggior parte delle istruzioni Z80 richiedono molto meno di un millisecondo per essere eseguite, quindi un programma qualunque durante la sua esecuzione genera normalmente tanti cicli di fetch tali di mantenere il rinfresco della memoria.

Le istruzioni più lunghe quali LDDR, CPIR, etc, sono fatte in modo tale che la memoria viene rinfrescata ad ogni trasferimento durante il corso dell'esecuzione. Quindi, i trasferimenti di alcune migliaia di byte non causano nessun problema per le memorie dinamiche.

Potete verificarlo voi stessi, realizzando un trasferimento di quel tipo.

Dovete però in questo caso a non scrivere nella zona di memoria del proprio programma e non distruggere lo spazio di memoria dei dati e dello 'stack' del sistema operativo, cioè da 0EF0 a 0FFF.

Passo 2

Un altro potenziale problema è quello dell'istruzione HALT. I progettisti della CPU Z80 hanno fatto in modo che la suddetta istruzione HALT sia equivalente all'esecuzione di un numero indefinito di istruzioni NOP.

Poichè una istruzione NOP richiede 4 cicli T, ed un ciclo T richiede circa 400 nanosecondi, quindi un NOP richiede circa 1,6 microsecondi, perciò molto meno di qualche millisecondo.

Per verificare sperimentalmente che l'istruzione di HALT non produce alterazioni nella memoria dinamica, caricate parecchie locazioni a partire dall'indirizzo 0300 con una configurazione regolare. Caricate la locazione 0220 con 00 (istruzione macchina per NOP) e la locazione 0221 con 76 (HALT). Mediante il tasto GO eseguire il programma nella locazione 0220. La CPU quindi rimarrà nello stato di HALT finchè non premerete il tasto BREAK, generando quindi un NMI. Controllate ora se i byte caricati precedentemente nella locazione 0300 e seguenti sono stati modificati oppure no.

Dovreste osservare che non sono stati modificati.

Passo 3

Ricablate il circuito di memoria per l'esperimento N. 4 in modo che la RAM statica 2101A sia nuovamente abbinata all'indirizzo 7F00-7FFF. Ricaricate il programma di test con inizio nella locazione MEM1. Eseguite il programma con inizio nella locazione XFER per copiare la routine di test della memoria MEM1 nella RAM statica. Accertatevi che il trasferimento sia avvenuto correttamente.

Passo 4

Con la routine MEM1, salvata nella RAM statica, azionate brevemente il segnale $\overline{BWAIT}$ della CPU Z80 verso massa, usate un filo per fare questo azionamento, e *non* un generatore di impulsi in quanto la linea $\overline{BWAIT}$ deve essere pilotata da una porta a collettore aperto.

Esaminate ora parecchie locazioni della memoria dinamica con inizio in MEM1. Cosa osservate?

Ciò che noi abbiamo visto non ha più nulla a che fare con quello che c'era prima. Vediamo ora cosa c'è a partire dalla locazione 7F00. Troverete esattamente la copia della routine MEM1.

Il passo 4 mostra un vantaggio importante per le memorie statiche rispetto a quelle dinamiche: mentre l'attivazione del segnale $\overline{BWAIT}$ per parecchi cicli di attesa, non ha effetto sulle RAM statiche, esso distrugge totalmente il contenuto delle RAM dinamiche.
Il motivo è che gli stati di attesa mettono la CPU in stato di STOP per l'esecuzione dei cicli di fetch in memoria. Quindi se vengono perduti troppi cicli di fetch, le memorie dinamiche non vengono più rinfrescate perdendo il loro contenuto.
Quando voi mettete a massa momentaneamente la linea $\overline{BWAIT}$, voi inserite decine di migliaia di cicli di attesa di 400 nanosecondi cadauno. Diecimila stati di attesa causano una interruzione di circa 4 millisecondi. Quindi, tenendo basso il segnale BWAIT per un tempo superiore si provoca la perdita del contenuto della RAM dinamica.
Notate che la perdita dell'informazione nella memoria avviene con una interruzione di 2 o 3 millisecondi, quindi l'inserimento di anche 1000 stati di attesa non può provocare effetti dannosi, ed è per questo motivo che le periferiche di I/O possono inserire degli stati di attesa senza alterare la configurazione della memoria.
Inoltre l'attività del segnale $\overline{BBUSRQ}$ sospende l'esecuzione dei cicli di fetch in memoria, in quanto il segnale suddetto fa si che tutti i segnali di controllo ed i bus siano assegnati ad una CPU esterna al Nanocomputer. Mentre la CPU esterna usa i bus del Nanocomputer, la CPU del Nanocomputer non può eseguire cicli di fetch in quanto questi richiederebbero loro stessi l'uso degli stessi bus.
Il $\overline{BBUSRQ}$ è utilizzato nelle configurazioni multi – CPU e per accessi diretti in memoria (DMA), due argomenti appena accennati in questo libro.

Passo 5

Poichè il programma di test della memoria è assegnato ad una memoria RAM statica, ora potete usarlo per testare la R/W RAM del vostro Nanocomputer.
Per testare le locazioni da 0100 a 01FF fare le seguenti modifiche:

Locazione	Vecchio contenuto	Nuovo contenuto
7F05	7F	01
7F0B	7F	01
7F0E	7F	01
7F16	7F	01
7F1D	01	7F

Eseguite ora il nuovo programma di test a partire dall'indirizzo 7F00.
In questo voi avete eseguito un programma residente in una R/W RAM, che voi avete personalmente aggiunto al vostro computer. Congratulazioni!

CAPITOLO 4

BUS, BUFFER THREE-STATE E I/O DELLO Z80

INTRODUZIONE

Come avete visto nei capitoli precedenti, la CPU dello Z80 comunica con l'esterno tramite tre bus: il bus dei DATI, il bus degli INDIRIZZI e il bus di CONTROLLO. In precedenza, abbiamo dato una breve definizione del termine *"bus"*. In questo capitolo, ne parleremo in modo dettagliato, esaminando la struttura del bus interno della CPU Z80, nonchè le tecniche relative all'impiego del bus, attualmente utilizzate, nella realizzazione di sistemi a microprocessori e minicomputers.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Dare la definizione del termine *bus* e del verbo omonimo (*to-bus*, in inglese).
- Descrivere le caratteristiche di un buffer TRI-STATE® o three-state, compresi gli ingressi di abilitazione/disabilitazione dei dati, nonchè l'uscita three-state.
- Scrivere una tabella della verità per un dispositivo three-state.
- Dare la definizione del termine *"gate a collettore aperto"* e capire come questi dispositivi vengano usati per realizzare i bus presenti nei sistemi a microcomputer.
- Fornire degli esempi di semplici sistemi di bus.
- Descrivere le caratteristiche generali dei dispositivi three-state, quali il 74LS125, il 74LS126 e il 74LS365.
- Fare un elenco di vari dispositivi three-state prodotti dalla SGS-ATES.
- Dare la definizione di *"fan-in"* e *"fan-out"* e spiegarne il significato relativamente ai circuiti di I/O dei microprocessori.
- Disegnare e montare un semplice circuito di uscita dello Z80.
- Disegnare e montare un semplice circuito d'ingresso dello Z80.
- Distinguere fra I/O relativo ad una porta e I/O memory mapped (in mappa di memoria) in un microcomputer Z80.

CHE COSA E' UN BUS?

Un *bus* è un insieme di canali comuni di conduzione sui quali vengono trasferite le informazioni digitali provenienti da una delle molte sorgenti e diretti verso una delle molte destinazioni. L'obiettivo fondamentale di un bus consiste nel minimizzare il numero di collegamenti necessari per trasferire le informazioni da un dispositivo digitale ad un altro. Di conseguenza, è possibile effettuare solo un trasferimento alla

volta. Mentre viene seguito tale trasferimento, tutte le altre sorgenti che sono collegate al bus devono essere disabilitate. Il verbo *(to bus)* significa: collegare fra di loro parecchi dispositivi digitali, che ricevono o trasmettono informazioni digitali, per mezzo di un insieme di canali comuni di conduzione, chiamato appunto bus e sul quale vengono trasferite tutte le informazioni scambiate fra un dispositivo e l'altro.

I bus sono presenti:

- all'interno dei circuiti integrati, ad esempio il bus dati interno al dispositivo microprocessore Z80;
- fra due circuiti integrati, ad esempio in un microcomputer Z80 i bus indirizzi, controllo e il bus dati bidirezionale presenti;
- fra i sistemi e gli strumenti digitali, ad esempio il bus di interfaccia IEEE-488, che costituisce attualmente un'interfaccia standard per strumenti digitali.

Il concetto di bus è probabilmente uno dei più importanti dell'elettronica digitale. Se non fossero in grado di condividere i canali di informazione, la maggior parte dei dispositivi digitali richiederebbero probabilmente un numero di collegamenti elettrici tre o quattro volte maggiore di quello attuale. I circuiti stampati dei microcomputer e dei minicomputer sarebbero notevolmente più complessi e, pertanto, più costosi e meno affidabili.

LA TECNICA DEL BUS THREE-STATE

Abbiamo detto in precedenza che un bus presenta queste due caratteristiche apparentemente contraddittorie:

- Unisce sorgenti multiple e destinazioni multiple tramite collegamenti comuni;
- In un preciso istante temporale è possibile effettuare, solo UN trasferimento di informazioni. Perciò, mentre è in corso sul bus un trasferimento di informazioni fra due dispositivi occorre sconnettere dal bus tutti i dispositivi che non sono coinvolti.

Prendiamo in considerazione il bus della Figura 4-1. I dispositivi TX1 e TX2 e i dispositivi RX1 e RX2 trasmettono e ricevono rispettivamente le informazioni sul bus. Se TX1 vuole trasmettere dei dati a RX1, e se allo stesso istante TX2 pone sul bus dei livelli logici 0 e 1, la trasmissione fra TX1 e RX1 verrà, in questo caso,

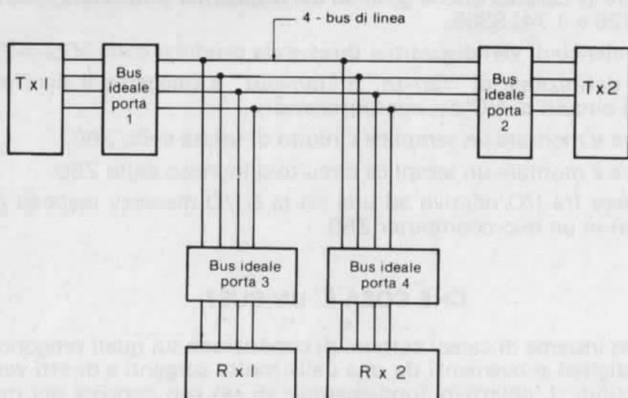


Figura 4-1. Un semplice sistema con quattro dispositivi e bus a quattro linee.

fortemente alterata. RX1 riceverà probabilmente un miscuglio di dati di TX1 e TX2. Chiaramente, TX2 deve essere in qualche modo disabilitato.
Si potrebbe togliere materialmente il collegamento elettrico tra TX2 e il bus, ma si tratta indubbiamente di una soluzione non realizzabile. La soluzione ottimale è che TX2, anzichè porre in uscita sul bus 0 ed 1 logici, assuma uno stato indicato come stato di "not here" (non presente). Perciò in un sistema con bus, la soluzione ottimale si raggiunge ponendo una porta fra il dispositivo e il bus stesso. Questa porta dovrebbe avere due stati di uscita digitale (livello logico 0 e 1, che rifletterebbero i dati del dispositivo), più un terzo stato corrispondente alla sconnessione o isolamento. La tabella della verità di questa porta ideale apparirebbe come nella Tabella 4-1.

Tabella 4-1. Tabella della verità per la porta ideale di un bus.

Dati in ingresso	Segnale di gating	Dati in uscita
0	abilitazione	0 (copia i dati del disp. sul bus)
1	abilitazione	1 (copia i dati del disp. sul bus)
0	disabilitazione	sconnette il dispositivo dal bus
1	disabilitazione	sconnette il dispositivo dal bus

Precedentemente, abbiamo parlato dei dispositivi digitali come dispositivi a due stati capaci di registrare un livello logico 0, equivalente al potenziale di massa, ed un livello logico 1, equivalente al potenziale +5V. Non abbiamo mai parlato di un terzo stato "di non collegato", che sembra fondamentale nell'implementazione di sistemi con bus. La prima a trovare una soluzione è stata la National Semiconductor Corporation, con lo sviluppo dell'uscita three-state o TRI-STATE® . Attualmente la SGS-ATES costruisce dispositivi three-state con le seguenti caratteristiche:

- Compatibilità con la serie 54/74 TTL
- Possibilità di collegare fino a 128 buffer ad una linea del bus
- Tempo di propagazione di 12 ns
- Capacità di pilotare elevati carichi capacitivi
- Controllo indipendente di ciascun buffer
- Permette di collegare insieme alle uscite rendendo, in tal modo, possibile il collegamento ad una sola linea del bus
- Miglioramento della integrità di forma d'onda e delle velocità rispetto a quella delle porte a collettore aperto (open collector)
- Miglioramento della comunicazione, nei due sensi, su una linea comune dato che nello stato di alta impedenza, gli ingressi del dispositivo three-state non caricano normalmente il dispositivo di pilotaggio.

Abilitazione/Disabilitazione	Ingresso	Uscita
0	0	Alta impedenza
0	1	Alta impedenza
1	0	0
1	1	1

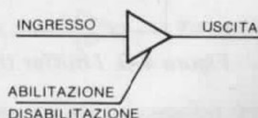


Fig. 4-2. Un buffer three-state abilitato da uno stato logico alto (High Enabled).

Riassumendo, un dispositivo *three-state* ha tre possibili stati di uscita:

Livello logico 0
Livello logico 1
Alta impedenza (non collegato)

Quando non è nello stato di alta impedenza, un dispositivo three-state si comporta come un normale dispositivo TTL; quando è disabilitato, si comporta come se non fosse collegato al circuito. Tutti i dispositivi three-state hanno un pin d'ingresso, chiamato di abilitazione/disabilitazione, che controlla se il dispositivo è in un normale stato TTL o in uno stato ad alta impedenza. La Figura 4-2 mostra lo schema e la tabella della verità di un buffer three-state abilitato da un ingresso a livello logico 1 sul suo pin di abilitazione/disabilitazione.

I BUFFER THREE-STATE QUADRUPLI 74LS125 E 74LS126

I buffer three-state quadrupli 74LS125 e 74LS126 contengono ciascuno quattro buffer ognuno dei quali è dotato di un proprio pin di abilitazione/disabilitazione. Il livello di abilitazione è l'unica differenza esistente tra i due dispositivi. I buffer 74LS125 sono abilitati con un livello logico 0, mentre i buffer 74LS126 sono abilitati con un livello logico 1. I due dispositivi sono illustrati nella Figura 4-3.

Per ogni dispositivo, gli ingressi di alimentazione corrispondono ai pin 7 e 14. L'accesso ai quattro buffer indipendenti avviene nel modo seguente:

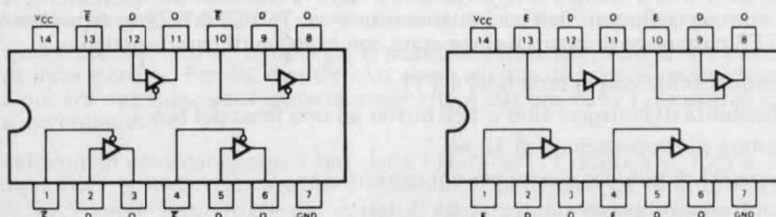


TABELLE DELLA VERITA'

LS125

INGRESSI Ē	D	USCITA
L	L	L
L	H	H
H	X	(Z)

LS126

INGRESSI E	D	USCITA
H	L	L
H	H	H
L	X	(Z)

L = livello di tensione basso (LOW)
H = livello di tensione alto (HIGH)
X = livello di tensione non significativo
(Z) = alta impedenza (OFF)

Figura 4-3. I buffer three-state quadrupli 74LS125 e 74LS126.

Buffer 1: Ingresso sul pin 2, uscita sul pin 3, abilitazione/disabilitazione sul pin 1.
Buffer 2: Ingresso sul pin 5, uscita sul pin 6, abilitazione/disabilitazione sul pin 4.
Buffer 3: Ingresso sul pin 9, uscita sul pin 8, abilitazione/disabilitazione sul pin 10.
Buffer 4: Ingresso sul pin 12, uscita sul pin 11, abilitazione/disabilitazione sul pin 13.

HEX BUFFER-STATE 74LS365 CON ABILITAZIONE COMUNE OTTENUTA TRAMITE NOR A 2 INGRESSI

Nella Figura 4-4 è illustrato l'HEX buffer 74LS365 insieme alla tabella della verità. Questo dispositivo contiene sei buffer che vengono abilitati simultaneamente dalla uscita di una porta NOR a 2 ingressi. Spesso nel collegamento dei dispositivi con i bus esterni dello Z80, non è necessario poter controllare separatamente ogni linea come per esempio quando si ricevono in ingresso o si inviano in uscita byte a otto bit. Per questa ragione il 74LS365 è un dispositivo molto usato.

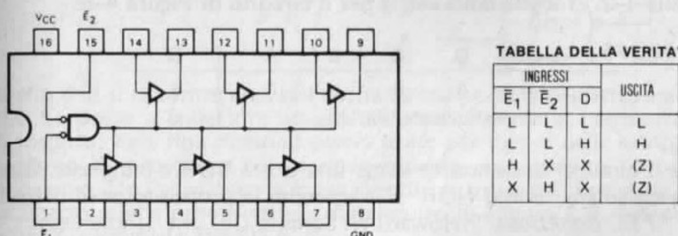


TABELLA DELLA VERITA'

INGRESSI			USCITA
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	L
L	L	H	H
H	X	X	(Z)
X	H	X	(Z)

Figura 4-4. Hex buffer 74LS365.

USCITE A COLLETTORE APERTO (OPEN COLLECTOR)

Un modo sempre meno diffuso, più lento, meno costoso e maggiormente soggetto a disturbi ("rumoroso") è quello di collegare insieme, su di un bus, le uscite di dispositivi logici, utilizzando le *uscite a collettore aperto*. Nei dispositivi a collettore aperto non è presente il resistore finale di "pull-up" del transistore di uscita del dispositivo perciò l'utente dovrà prevedere questo resistore prima di completare il suo circuito.

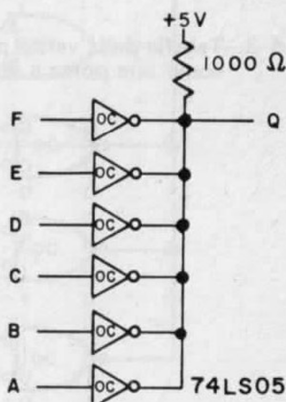


Figura 4-5. Invertitore esadecimale 74LS05 con le sei uscite collegate insieme a +5V utilizzando un solo transistore di pull-up da 1000 Ohm.

Ad ogni modo, dato che il resistore di pull-up manca in tutti i dispositivi logici con uscite a collettore aperto, è possibile collegare insieme tutte queste uscite ed usare per tutte un solo resistore di pull-up. Per esempio, per costruire il circuito di Figura 4-5, è possibile usare i sei invertitori con uscite a collettore aperto, presenti nello invertitore esadecimale 74LS05.

Sul dispositivo 74LS05 le sei uscite degli invertitori sono collegate insieme a +5V usando un solo resistore di pull-up di 1000 Ohm. L'uscita, Q, è ad uno stato logico 1 solo se ABCDEF = 000000. Se uno qualunque dei sei ingressi si trova nello stato logico 1, l'uscita di un invertitore di questo tipo è in uno stato logico 0 e mette a massa o "tira giù" (Pulls down), le uscite dei rimanenti cinque invertitori.
In queste condizioni l'uscita Q, si trova nello stato logico 0. La tabella della verità di questo circuito è riportata nella Tabella 4-2.

Tabella 4-2. Tabella della verità per il circuito di Figura 4-5.

F	E	D	C	B	A	Q
0	0	0	0	0	0	1
tutti gli altri stati						0

E' chiaro che il circuito si comporta come una porta NOR a 6 ingressi. Questa porta viene talvolta chiamata "wired NOR". Un esempio in proposito ce lo offre Lancaster a p. 138 del *"TTL Cookbook"* (Howard W. Sams & Co., Inc., Indianapolis, Indiana).

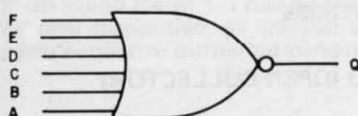


Figura 4-6. Porta NOR a sei ingressi.

La Figura 4-6 mostra una porta NOR a 6 ingressi. Quando si collegano insieme le uscite a collettore aperto di una coppia di porte NAND a 2 ingressi si ottiene il circuito di Figura 4-7.
La tabella della verità è data nella Tabella 4-3.

Tabella 4-3. Tabella della verità per il circuito di figura 4-7. Il circuito si comporta come una porta a due ingressi e due livelli AND-OR-INVERTITORE.

D	C	B	A	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

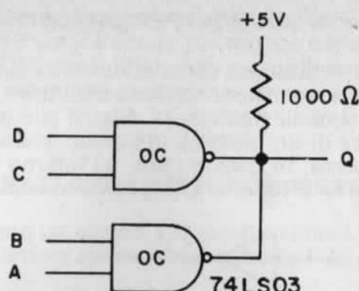


Figura 4-7. Coppia di porte NAND a 2 ingressi con uscite Open Collector collegate insieme.

Dalla Tabella 4-3 si riscontra che, se l'uscita di una porta o dell'altra è a livello logico 0, Q = 0. Le porte a collettore aperto collegate insieme si comportano in modo piuttosto insolito. Esse non possono essere usate per creare delle semplici porte! Vi raccomandiamo di procedere con cautela quando usate molte porte a 2 ingressi collegate insieme. Una eccezione a questa regola si trova nell'uso della porta AND a collettore aperto, come potete vedere dal circuito di Figura 4-8, e di cui riportiamo la tabella della verità nella Tabella 4-4.

Tabella 4-4. Tabella della verità per il circuito di Figura 4-8.

H	G	F	E	D	C	B	A	Q
1	1	1	1	1	1	1	1	1
tutti gli altri stati								0

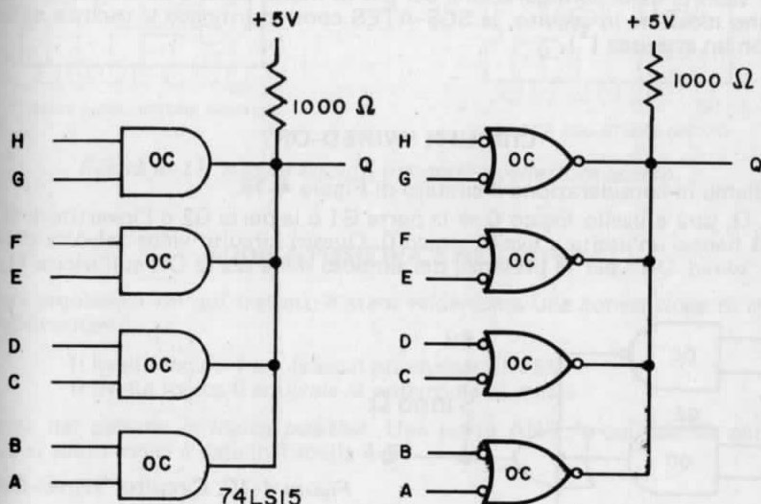


Figura 4-8. Due rappresentazioni equivalenti di quattro porte AND a 2 ingressi con uscite collegate insieme.

Osservate che la seconda rotazione utilizza dei simboli che non appaiono come porte AND. In una sezione successiva di questo capitolo discuteremo questo tipo di rappresentazione.

SIMBOLI PER DISPOSITIVI A COLLETTORE APERTO

Non vi abbiamo detto molto circa la standardizzazione dei simboli che si applicano alle porte contenenti uscite a collettore aperto. Nella maggior parte dei casi, vengono usati i simboli tradizionali relativi alle porte e sta a voi dedurre, dallo schema o dalla presenza di un resistore di pull-up, che in quel caso vengono impiegate porte a collettore aperto. In questo libro, all'interno degli appropriati simboli per le porte logiche inseriremo le lettere "OC", come mostrato nella Figura 4-9.

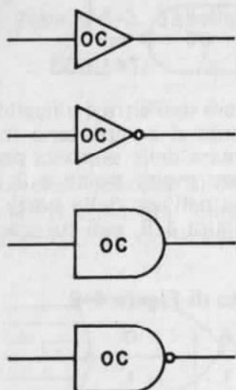


Figura 4-9. Simboli per i dispositivi logici a collettore aperto.

Come vedrete dalla configurazione dei pin di molte porte a collettore aperto, che vi verranno mostrate in seguito, la SGS-ATES contraddistingue le uscite a collettore aperto con un asterisco (*).

CIRCUITI "WIRED-OR"

Prendiamo in considerazione il circuito di Figura 4-10.
L'uscita, Q, sarà a livello logico 0 se la porta G1 o la porta G2 o l'invertitore G3 o la porta G4 hanno un'uscita a livello logico 0. Questo circuito viene talvolta chiamato circuito "wired-OR" per la presenza del simbolo della porta OR sull'uscita Q. Rite-

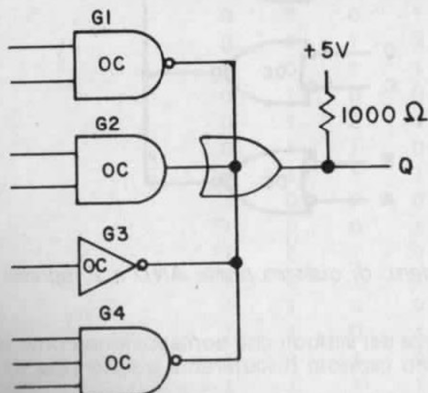
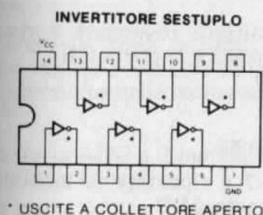


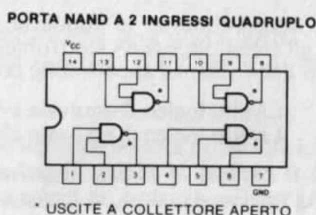
Figura 4-10. Circuito "Wired-OR"

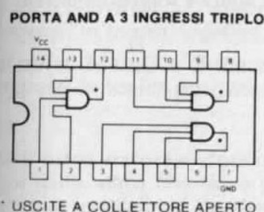
niamo che l'uso di questo termine possa trarre in inganno, poichè il circuito non si comporta affatto come una semplice porta OR a più ingressi, per quanto concerne gli ingressi della porta. Piuttosto la funzione OR viene implementata sull'uscita delle porte. State sempre molto attenti quando incontrate il termine "wired-OR" nella letteratura elettronica.

ESEMPI TIPICI DI CIRCUITI INTEGRATI A COLLETTORE APERTO

In Figura 4-11 sono riportati per riferimento un elenco, e le configurazioni dei pin ad essi relative, di alcuni circuiti integrati a collettore aperto, prodotti dalla SGS-ATES.







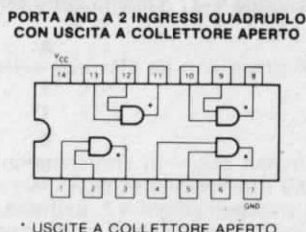


Figura 4-11. Alcuni circuiti integrati a collettore aperto.

LOGICA POSITIVA E NEGATIVA

Negli argomenti fin qui trattati, è stata evidenziata una convenzione di importanza fondamentale:

Il livello logico 1 equivale al potenziale di +5V
Il livello logico 0 equivale al potenziale di massa

Si tratta del *sistema in logica positiva*. Una porta AND la cui tabella della verità relativa ai valori logici è data in Tabella 4-5:

Tabella 4-5. Tabella logica della verità per una porta AND a logica positiva.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

e il cui comportamento elettrico è:

Tabella 4-6. Comportamento elettrico di una porta AND a logica positiva.

A	B	Q
GND	GND	GND
GND	+5V	GND
+5V	GND	GND
+5V	+5V	+5V

si chiama porta AND *in logica positiva.*

Che cosa accadrebbe se le suddette equivalenze venissero rovesciate e dovessero rimanere gli stessi dispositivi elettronici come la porta AND e la porta OR?
In tal caso l'equivalenza apparirebbe così:

Livello logico 0 equivale a +5V
Livello logico 1 equivale al potenziale di massa.

Questo è il *sistema in logica negativa*. In Tabella 4-7 è riportata la tabella *logica* della verità relativa ai valori, in *logica negativa* di una porta AND.

Tabella 4-7. Tabella della verità per una porta AND a logica negativa.

A	B	Q*
1	1	1
1	0	1
0	1	1
0	0	0

La tabella è identica a quella relativa alla porta OR in *logica positiva*. Perciò, se vi trovate a lavorare su circuiti che impiegano la logica negativa, e viene richiesta una porta AND (logica negativa), il dispositivo da usare in questo caso dovrebbe essere una porta OR quadrupla a due ingressi 74LS32 (logica positiva). In realtà non è così confuso come può sembrare.
Per trovare l'equivalente in logica negativa di un dato dispositivo in logica positiva, basta scambiare gli 0 e gli 1 della tabella della verità del dispositivo in logica positiva, osservare bene il risultato e riconoscere la porta corretta. Per esempio, qual'è l'equivalente in logica negativa di una porta NAND, in logica positiva a due ingressi?
Per rispondere a questa domanda si proceda come segue:

Passo 1: Scrivere la tabella della verità di una porta NAND in logica positiva:

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

Passo 2: Scambiare fra di loro gli 0 e gli 1 per formare una tabella della verità in logica negativa (l'asterisco (*) dopo l'uscita Q indica che la tabella si riferisce ad un dispositivo in logica negativa).

A	B	Q *
1	1	0
1	0	0
0	1	0
0	0	1

Passo 3: Cercate di riconoscere la porta rappresentata in questa tabella. Questa tabella corrisponde a quella di una porta NOR.

Perciò una porta NAND in logica positiva equivale ad una porta NOR in logica negativa.

Applicando la stessa tecnica, è possibile verificare che una porta NAND in logica negativa equivale ad una porta NOR in logica positiva. Riportiamo di seguito un elenco completo delle equivalenze:

Una porta AND a 2 ingressi in logica positiva equivale ad una porta OR a 2 ingressi in logica negativa.

Una porta AND a 2 ingressi in logica negativa equivale ad una porta OR a 2 ingressi in logica positiva.

Una porta NAND a 2 ingressi in logica positiva equivale ad una porta NOR a 2 ingressi in logica negativa.

Una porta NAND a 2 ingressi in logica negativa equivale ad una porta NOR a 2 ingressi in logica positiva.

E' del tutto naturale domandarsi perchè la convenzione in logica negativa viene sempre a galla, dato che (a) è poco chiaro (b) le sue esigenze concernenti l'hardware vengono già soddisfatte da dispositivi in logica positiva. La logica negativa è spesso assai utile nella semplificazione di complicati circuiti in logica positiva. La ragione di base è strettamente legata alla natura dei circuiti a collettore aperto. Prima di tutto, nei circuiti a collettore aperto, i livelli logici zero sono molto più facilmente riconoscibili o "elettricamente visibili". In un sistema con dei bus in cui le uscite di più dispositivi sono collegate insieme è sufficiente che l'uscita di un componente sia a livello logico zero perchè l'uscita comune vada anch'essa a livello logico zero. Questo ci permette di "evidenziare" la presenza di livelli logici zero rispetto alla presenza di livelli logici 1. Generalmente, nei sistemi a logica positiva, ci si basa su questa considerazione:

"dati" = 1 e "assenza dati" = 0

Ovvero, una linea in cui stanno per essere trasmesse delle informazioni digitali, si trova nel suo stato quiescente con potenziale di massa, e il verificarsi di un impulso a +5V stà ad indicare il flusso dei dati lungo la linea. Nei circuiti a collettore aperto, ciò che si evidenzia è il verificarsi di livelli logici 0. Quindi, è naturale considerare la trasmissione dei dati nel contesto dei circuiti a collettore aperto, come segue:

"dati" = 0 e "assenza dati" = 1

Lo stato di quiescenza di una linea, alla quale sono collegate molte uscite a collettore aperto, è a livello logico 1, ed "un'azione" viene identificata con l'abbassamento di tutta la linea da parte di una porta. Il diagramma delle temporizzazioni di Figura 4-12 illustra la trasmissione della parola digitale a sette bit, 1010101, su una linea positiva ed una negativa.

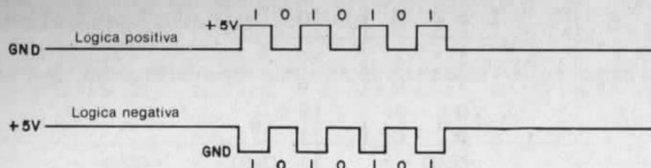


Figura 4-12. Diagramma delle temporizzazioni in logica positiva e negativa relativa alla dimensione della parola 1010101.

NOTAZIONI DI LOGICA POSITIVA E DI LOGICA NEGATIVA

Nei precedenti capitoli, e nella Figura 4-8 di questo capitolo, avete osservato i piccoli cerchi agli ingressi ed alle uscite delle porte e dei circuiti integrati. Alcuni esempi sono riportati in Figura 4-13.

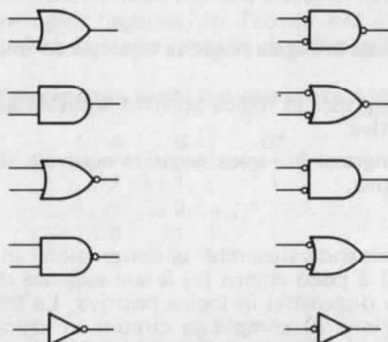


Figura 4-13. Esempio di utilizzazione dei cerchietti di inversione.

Questi piccoli cerchi sono spesso detti *cerchi di inversione* poichè possono essere considerati come una notazione semplificata di un invertitore.
Un metodo, egualmente importante, per interpretarli è legato al concetto di logica positiva e negativa che abbiamo appena visto. Ad esempio, in Tabella 4-8 è riportata una rappresentazione simbolica delle equivalenze tra porte in logica positiva e negativa.

Vediamo una tipica applicazione di queste equivalenze nell'interfacciamento dello Z80. Nella CPU Z80 la maggior parte dei segnali di *controllo* sono attivi con livello logico basso e, di conseguenza, sono segnali in logica negativa. Allora, se volete creare un impulso *negativo* solo e solo se $\overline{IORQ}$ *e* $\overline{RD}$ sono attivi, potete usare una porta AND in logica negativa, come mostrato in Figura 4-14.



Figura 4-14.

In realtà, la porta che userete sarà una porta OR in logica positiva, ad esempio il dispositivo 74LS32-OR quadruplo a due ingressi.
Quello che dovete imparare progettando il circuito è: come prima cosa scegliere la funzione logica che vi serve, in questo caso un AND, poi disegnate il simbolo ed infine pensate secondo la notazione *attiva*:
"Se $\overline{IORQ}$ e $\overline{RD}$ sono *attivi* allora desidero generare un segnale di uscita *attivo"*. Se *attivo* significa *basso*, occorre inserire i simboli di inversione. In questo caso, *attivo* significa *basso* per entrambi gli ingressi e per l'uscita. Vediamo ora il caso in cui il problema viene modificato: generare un impulso positivo solo e solo se $\overline{IORQ}$ e $\overline{RD}$ sono attivi. Allora, dato che sapete che $\overline{IORQ}$ e $\overline{RD}$ sono attivi sul *basso*, e che un impulso positivo è attivo sull'*alto*, dovreste progettare il circuito illustrato in Figura 4-15.

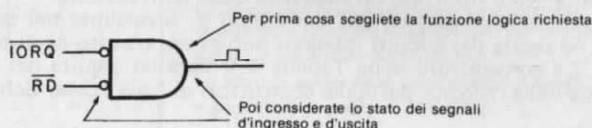


Figura 4-15.

Tabella 4-8. Tabella delle equivalenze e dei simboli relativi alle porte in logica negativa e positiva.

Funzione in logica positiva	Simbolo della porta in logica positiva	Simbolo della porta in logica negativa	Definizione dell'equivalenza
OR			Una porta OR in logica positiva è equivalente ad una porta AND in logica negativa
AND			Una porta AND in logica positiva è equivalente ad una porta OR in logica negativa
NOR			Una porta NOR in logica positiva è equivalente ad una porta NAND in logica negativa
NAND			Una porta NAND in logica positiva è equivalente ad una porta NOR in logica negativa
INV			Un invertitore in logica positiva è equivalente ad un invertitore in logica negativa

Questo circuito richiede una porta NAND in logica negativa, o una porta NOR in logica positiva. Perciò, nella vostra scatola di componenti cercherete un dispositivo 74LS02. Per esempi più sofisticati, di utilizzo dei simboli delle porte negative nei circuiti di interfaccia per le CPU Z80, guardate gli schemi del Capitolo 3, relativi agli impulsi di selezione dispositivo. Per tutto il resto di questo libro useremo i precedenti simboli di porte.
Occorre ricordare la seguente convenzione: I SEGNALI ATTIVI SUL BASSO SI CONSIDERANO SEGNALI DI LOGICA NEGATIVA ED I CERCHI DI INVERSIONE SONO USATI IN BASE A QUANTO SEGUE: I CERCHI DI INVERSIONE SONO ASSOCIATI AGLI INGRESSI ATTIVI SUL BASSO E ALLE USCITE ATTIVE SUL BASSO.

Tali convenzioni si applicano agli ingressi ed alle uscite di dispositivi e circuiti digitali. In Tabella 4-9 è riportato un riassunto delle convenzioni.
Inoltre, un eccellente esempio di uso dei cerchi di inversione nel contrassegnare i pin di ingresso ed uscita dei circuiti integrati può essere trovato negli schemi riportati nel Capitolo 3. Le convenzioni della Tabella 4-9 saranno seguite nel resto di questo libro come anche nella maggior parte dei costruttori di Low Power Schottky TTL, Ics.

BUS NEGATIVI E POSITIVI

Parlando di logica positiva e negativa, conviene distinguere fra *bus positivo* e *bus negativo*. Su di un bus positivo lo stato quiescente corrisponde a GND, mentre su di un bus negativo, lo stato quiescente corrisponde a +5V. La logica three-state comporta dei sistemi costituiti da bus positivi, mentre la logica a collettore aperto comporta sistemi sostituiti da bus negativi.

Nello schema di Figura 4-16, potete vedere un bus negativo al quale sono collegati quattro dispositivi logici con uscite a collettore aperto. Lo stato quiescente del bus è a +5V. Se l'uscita di *una* qualunque delle porte va a livello logico 0, (o massa), la linea del bus si abbassa con essa. Se una qualunque delle porte che leggono i dati dal bus (porte G, F, E) è abilitata, essa è in grado di sentire la variazione dello stato logico.

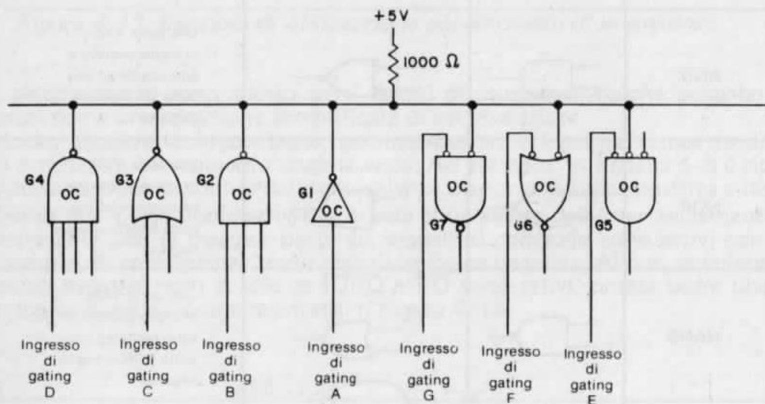


Figura 4-16. Bus a logica negativa.

Termina qui la nostra discussione sui bus negativi e la logica negativa. Pertanto, tutti i riferimenti alla logica ed ai bus dovrebbero essere considerati positivi, a meno che non venga specificato diversamente.

Tabella 4-9. Significato dei potenziali per i cerchi di inversione agli ingressi ed alle uscite di circuiti e dispositivi digitali.

Simbolo	Convenzione
Ingressi	Un 1 logico abilita il dispositivo od il circuito. Un 0 logico disabilita il dispositivo od il circuito. Un fronte positivo costituisce l'abilitazione, lo strobe, il trigger od il clock del dispositivo o del circuito. Un impulsivo positivo di clock costituisce lo strobe od il clock del dispositivo o del circuito. Il dato è presentato non invertito. Lo stato di quiescenza dell'ingresso è basso. Il segnale di ingresso è attivo alto.
	Uno 0 logico abilita il dispositivo ed il circuito. Un 1 logico disabilita il dispositivo ed il circuito. Un fronte negativo costituisce la abilitazione, lo strobe, il trigger od il clock del dispositivo o del circuito. Un impulsivo negativo di clock costituisce lo strobe od il clock del dispositivo o del circuito. I dati sono presentati invertiti. Lo stato quiescente dell'ingresso è alto. Il segnale di ingresso è attivo basso.
Uscite	L'Uscita dal dispositivo o dal circuito non è invertita. L'uscita dal dispositivo o dal circuito è un impulso di clock positivo. Lo stato quiescente dell'uscita è basso. Il segnale di uscita è attivo alto.
	L'uscita dal dispositivo o dal circuito è invertita. L'uscita dal dispositivo o dal circuito è un impulso di clock negativo. Lo stato quiescente dell'uscita è alto. Il segnale di uscita è attivo basso.

CIRCUITO BUFFER/LATCH

Attualmente, la tecnologia dominante riguardante i bus, cioè quella usata nella maggior parte dei microprocessori, è quella three-state a logica positiva. La Figura 4-17 mostra una configurazione circuitale molto comune che viene usata sia allo interno dei microprocessori sia per collegamenti esterni a questi nel buffer/latch three-state.

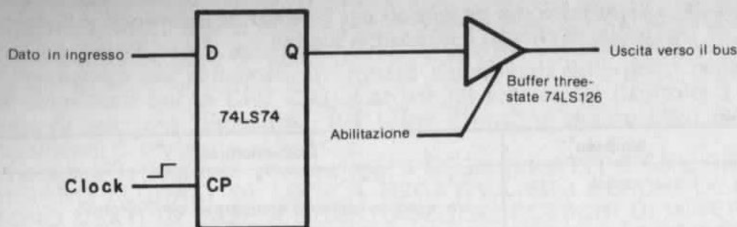


Figura 4-17. Circuito buffer/latch three-state.

Il flip-flop 74LS74 si comporta come un elemento di memoria, mentre il buffer three-state controlla il "collegamento" fra l'uscita del flip-flop e il bus. La tabella della verità ad esso relativa è data in Tabella 4-10.

Con questo tipo di circuito di uscita, è possibile conservare i dati di uscita in un latch e porli sul bus in un secondo tempo. Questo particolare può risultare molto importante nel caso di contenzione del bus, cioè se vi sono più richieste contemporanee di bus.

Tabella 4-10. Tabella della verità per il circuito del latch three-state di Figura 4-14.

Clock	Abilitazione	Condizione di uscita
0	0	Viene effettuato un latch sui dati precedenti: l'uscita three-state è disabilitata
0	1	Viene effettuato un latch sui dati precedenti, che sono posti in uscita sui bus
0-1-0	0	Il latch segue l'ingresso dei dati; l'uscita three-state è disabilitata quindi i nuovi dati non vengono posti in uscita sul bus
0-1-0	1	Il latch segue l'ingresso dei dati, immediatamente posti in uscita sul bus. Comportandosi come un semplice latch, i dati inseriti vengono immediatamente riflessi sull'uscita del circuito.

ESEMPI DI SISTEMI DI BUS POSITIVI

Nella Figura 4-18, è illustrato un semplice bus positivo ad una linea, con quattro dispositivi, basato sull'uso di un solo buffer three-state 74LS126. Questo circuito è identificato come bus dato che *sono collegate insieme le uscite delle porte da A a D.* Con i dispositivi standard TTL della serie 7400, questo non è possibile, a meno che i dispositivi non abbiamo speciali circuiti di uscita del tipo che permettano di realizzare un bus three-state o a collettore aperto.

Il funzionamento del circuito dovrebbe risultare chiaro se assumiamo che le porte da A a D della Figura 4-18 vengono abilitate da un segnale di livello logico 1. E' sempre possibile in un generico istante abilitare una *sola porta del buffer; le rimanenti porte devono essere disabilitate.* Perciò, sul bus ad una sola linea, è possibile vedere apparire, in qualunque momento, le informazioni digitali provenienti da uno solo dei quattro buffer. Le informazioni provenienti dagli altri tre buffer vengono bloccate, dato che i buffer corrispondenti sono disabilitati. La Tabella della verità 4-11 descrive il funzionamento del circuito.

Tabella 4-11. Tabella della verità per il circuito riportato in Figura 4-18.

D	C	B	A	Uscita
0	0	0	0	Nessun dispositivo abilitato
0	0	0	1	QA (I buffer B, C e D sono scollegati dal bus)
0	0	1	0	QB (I buffer A, C e D sono scollegati dal bus)
0	1	0	0	QC (I buffer A, B e D sono scollegati dal bus)
1	0	0	0	QD (I buffer A, B e C sono scollegati dal bus)

E' importante notare che: *Per questo circuito tutte le altre condizioni logiche sono considerate "illegali" poichè fanno apparire, sul bus ad una sola linea, le informazioni provenienti da più di un buffer. Se tentate di implementare una di queste condizioni d'ingresso "illegali", molto probabilmente brucerete il dispositivo three-state!*

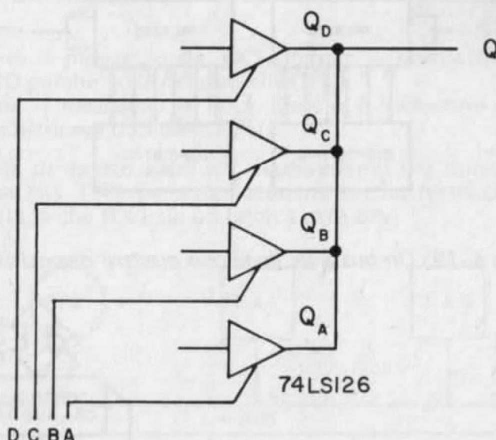


Figura 4-18. Un bus positivo ad una linea con quattro dispositivi.

Molti sistemi utilizzano bus a più linee, come mostra la Figura 4-19. A parte il fatto che gli ingressi di gating abilitano o disabilitano contemporaneamente le sei porte del buffer, quindi viene usato il chip 74LS365, questo circuito è concettualmente identico a quello mostrato nella Figura 4-18. Se supponiamo che gli ingressi A, B C e D siano collegati a entrambi gli ingressi della porta NOR a due ingressi, che controlla i sei ingressi di abilitazione/disabilitazione, ponendo un livello logico 0 su uno qualunque di questi ingressi si abilita l'uscita a sei bit. La tabella della veriá relativa a questo circuito è data nella Tabella 4-12.

Tabella 4-12. Tabella della verità per il circuito di Figura 4-19.

D	C	B	A	Uscita
1	1	1	1	Nessun dispositivo abilitato
1	1	1	0	Dispositivo A — sei bit paralleli
1	1	0	1	Dispositivo B — sei bit paralleli
1	0	1	1	Dispositivo C — sei bit paralleli
0	1	1	1	Dispositivo D — sei bit paralleli

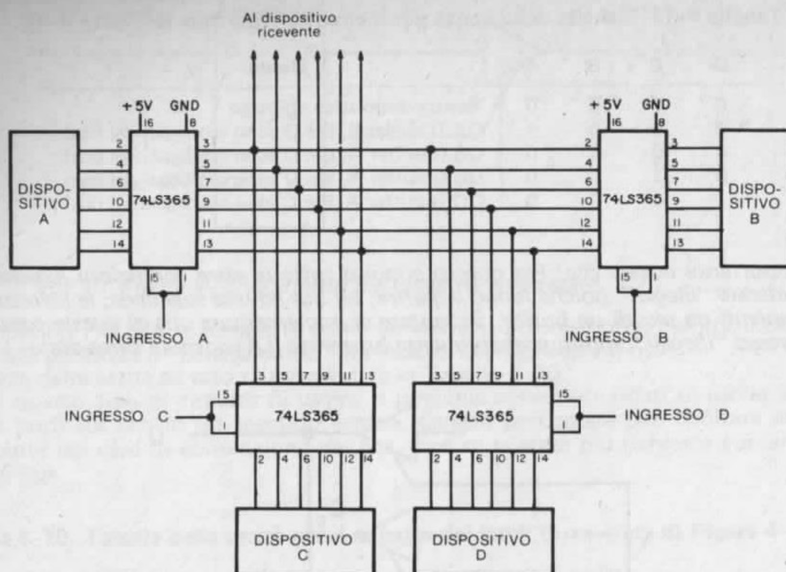


Figura 4-19. Un bus a sei linee con quattro dispositivi.

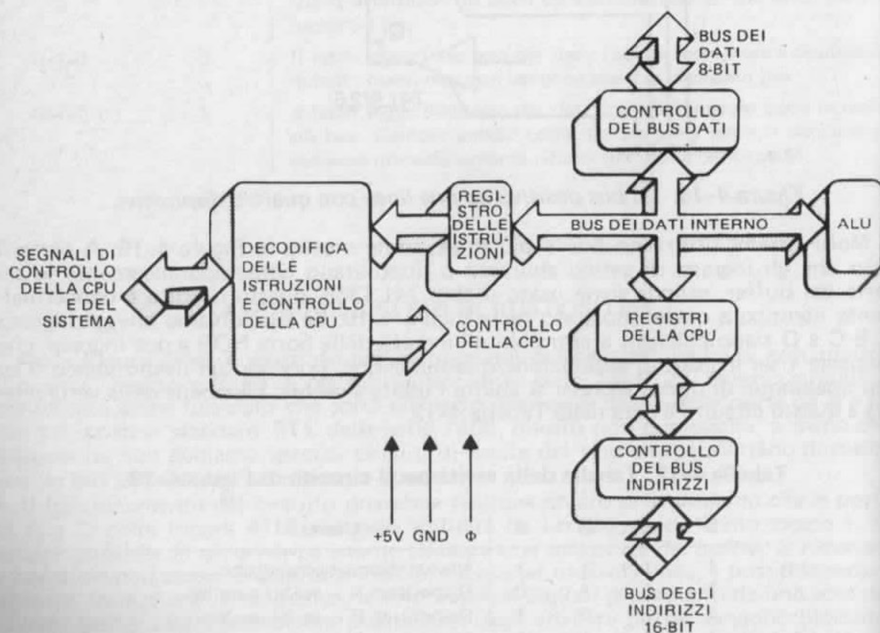


Figura 4-20. Il bus dei dati interno della CPU Z80 e blocchi funzionali.

LA STRUTTURA DEL BUS INTERNO ALLA CPU Z80

Oltre ai bus esterni (indirizzi, controllo e dati) la CPU Z80 ha un bus dei dati interno costituito da quattro dispositivi e otto linee. La Figura 4-20 illustra il bus dei dati interni e la correlazione esistente fra questo bus e i principali elementi funzionali della CPU Z80, e fra questo e i bus esterni.
Anche se il bus di dati interno alla CPU dello Z80 è implementato all'interno di un solo circuito integrato, le regole che ne governano l'accesso non sono diverse da quelle date per i bus discussi precedentemente in questo capitolo.

INGRESSO E USCITA DEI MICROCOMPUTER IN UN SISTEMA A BUS

Prendiamo in considerazione nella Figura 4-21, un bus con una CPU, due dispositivi trasmittenti TX1 e TX2, e due dispositivi riceventi RX1 e RX2. Supponiamo che TX1 voglia inviare un messaggio a RX1 tramite la CPU. In tal caso devono verificarsi i seguenti passi:

1. La CPU riceve il messaggio da TX1 (Perciò, il trasmettitore è un dispositivo di INGRESSO perchè invia dei dati alla CPU).
2. La CPU invia il messaggio a RX1 (Perciò il ricevitore è un dispositivo di USCITA perchè riceve dati dalla CPU.

Nella parte restante di questo capitolo, esamineremo nei minimi particolari i passi 1 e 2. Supporremo che TX1 sia semplicemente un buffer/latch a otto bit con un messaggio di un byte, e che RX1 sia un latch a otto bit.

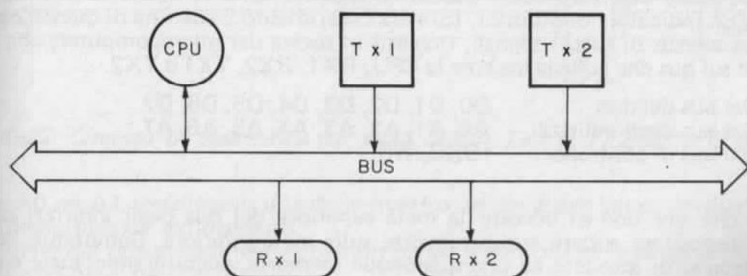


Figura 4-21. Un bus con una CPU, due trasmettitori TX1 e TX2, e due ricevitori RX1 e RX2.

Perchè anche RX1 non è un buffer/latch? Questa è una domanda importante a cui risponderemo più avanti. Esaminiamo prima l'uscita del microcomputer, ovvero il Passo 2.

L'USCITA DEL MICROCOMPUTER

Supponiamo che la CPU abbia nel suo accumulatore il messaggio, ad un solo byte, per RX1. Che hardware e che software sono necessari per inviare il byte sul bus verso RX1? Quante linee deve avere il bus? Quali segnali dovrebbero essere inviati? Come farà RX1 a sapere quando il byte del messaggio si trova sul bus? Ora siamo pronti a rispondere a tutte queste domande.

Nel Capitolo precedente abbiamo parlato del gruppo di istruzioni di I/O. In particolare, abbiamo detto che l'istruzione OUT A,(n) faceva sì che sui bus degli indirizzi dei dati e di controllo si verificasse quanto segue:

1. La CPU pone il codice dispositivo "n" sulla metà inferiore (A0–A7) del bus degli indirizzi, e il contenuto dell'accumulatore sulla metà superiore (A8–A15) del bus degli indirizzi.
2. La CPU pone il contenuto dell'accumulatore sul bus dei dati.
3. La CPU attiva simultaneamente i segnali $\overline{IORQ}$ e $\overline{WR}$.
4. La CPU disattiva i segnale $\overline{IORQ}$ e $\overline{WR}$.

Come può tutto ciò avere attinenza con il nostro particolare problema? Ecco la risposta: le informazioni, associate ai punti da 1 a 4, possono essere unite insieme a formare un *impulso di selezione dispositivo* che viene usato per due scopi importanti:

1. Esso seleziona RX1 e non RX2 come ricevitore dei dati, inviando impulsi di strobe a RX1 e non a RX2.
2. Esso notifica a RX1 che il messaggio ad un byte si trova in quel momento sul bus dei dati.

Dato che le temporizzazioni sono tali che l'impulso di selezione dispositivo arriva subito dopo che i dati dell'accumulatore sono stati posti sul bus di dati, l'impulso di selezione dispositivo può essere usato per abilitare il latch a otto bit e RX1 a ricevere i dati. Il risultato è che, subito dopo che i dati dell'accumulatore si sono posizionati sul bus dei dati, l'impulso di selezione dispositivo temporizza i flip-flop di RX1, ed il messaggio a otto bit, in attesa su ognuno degli ingressi D, viene copiato sulle uscite Q. E' possibile osservare tutto questo collegando tutte le uscite Q dei flip-flop di RX1 ad un indicatore di monitor. (Si veda Esperimento 5 alla fine di questo capitolo). Ecco un elenco di tutti i segnali, richiesti in uscita dal microcomputer, che devono apparire sul bus che collega insieme la CPU, RX1, RX2, TX1 e TX2.

Dal bus dei dati	D0, D1, D2, D3, D4, D5, D6, D7
Dal bus degli indirizzi	A0, A1, A2, A3, A4, A5, A6, A7
Dal bus di controllo	$\overline{IORQ}$, $\overline{WR}$.

Notate che ora non ci occorre la metà superiore del bus degli indirizzi perchè il codice dispositivo appare esclusivamente sulla metà inferiore. Comunque, se mai ci fosse bisogno di accedere ad una qualunque memoria, occorreranno tutte e sedici le linee d'indirizzamento. Perciò, dato che non vogliamo venire a trovarci in una situazione che non presenta vie di uscita, aggiungiamo al nostro bus i seguenti segnali:

Dal bus degli indirizzi:	A8, A9, A10, A11, A12, A13, A14, A15
Dal bus di controllo:	$\overline{MREQ}$, $\overline{RD}$

Con questi segnali e con un'appropriata circuiteria di decodifica, possiamo leggere e scrivere sia nei dispositivi di I/O che nelle locazioni di memoria.
Chiaramente non tutti i dispositivi saranno materialmente collegati con tutte le linee del bus. Ad esempio, RX1 ed i circuiti di selezione dispositivo ad esso associati, saranno collegati solo al primo gruppo di segnali elencati più sopra.
In effetti, si potrebbe usare la stessa circuiteria di selezione del dispositivo per fornire gli impulsi di strobe a RX1 e RX2, o addirittura a tutti e quattro i dispositivi, RX1, RX2, TX1 e TX2. Questa possibilità è illustrata nella Figura 4-22.

I circuiti necessari a generare l'impulso di selezione dispositivo utilizza un 74LS155, doppio decodificatore/demultiplexer 1 su 4, abilitato dall'uscita di un circuito "wired-OR" avente come ingresso le linee di indirizzo A2–A7.
Il 74LS155 è abilitato se e solo se le linee di indirizzo A7 ed A6 sono nello stato logico 1, e le linee A2, A3, A4 ed A5 sono in quello 0.

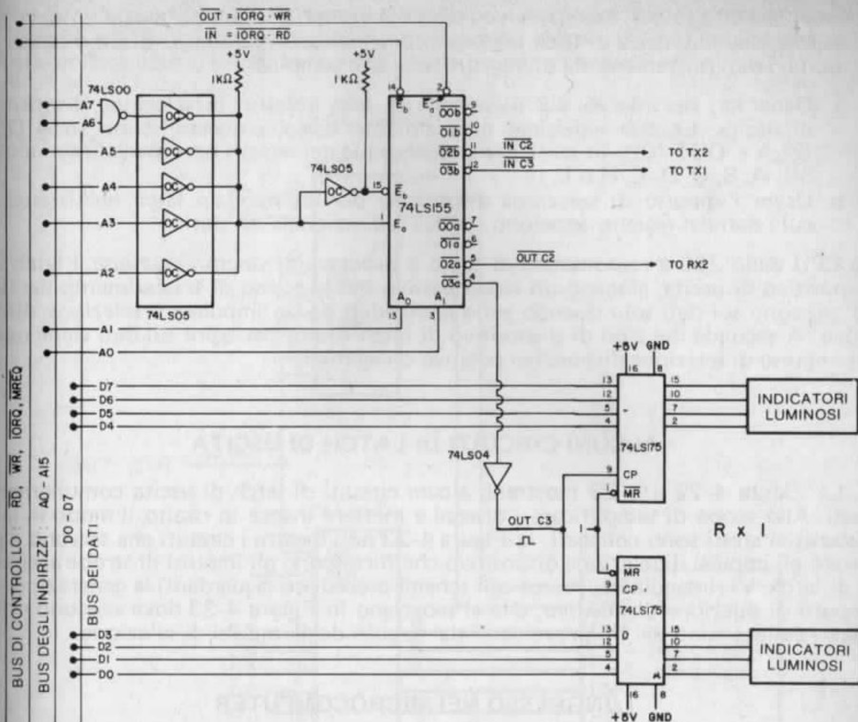


Figura 4-22. Circuito di decodifica per RX1, RX2, TX1 e TX2; schema per RX1.

Le linee A0 ed A1 selezionano una delle quattro uscite, attive basse, dei due decodificatori/demultiplexer indipendenti.

L'attivazione dei due decodificatori/demultiplexer è controllata rispettivamente dagli ingressi Ēa ed Ēb ai pin 2 e 14.

Dato che Ēa è connesso al segnale, attivo basso, generato quando entrambi $\overline{IORQ}$ e $\overline{WR}$ sono bassi, il decodificatore/demultiplexer "a" è riservato ad operazioni di uscita.

Come potete vedere, le sue uscite sono usate per lo strobe dei RICEVITORI RX1 ed RX2.

Similarmente, dato che Ēb è connesso con il segnale, attivo basso, generato dalla attivazione contemporanea di $\overline{IORQ}$ e $\overline{RD}$, il decodificatore/demultiplexer "b" è dedicato ad operazioni di ingresso, e, quindi, le sue uscite attivano lo strobe dei TRASMETTITORI TX1 e TX2.

Il codice dispositivo sugli otto bit inferiori del bus degli indirizzi è decodificato in *assoluto* in modo da determinare quale delle linee di uscita è attivata.

Oltre ad una implementazione leggermente differente di un vecchio concetto, gli unici concetti nuovi presentati nel circuito della Figura 4-22 consistono nella rappresentazione "bus oriented" delle linee d'indirizzamento, dei dati e di controllo provenienti dalla CPU, oltre ai due latch a quattro bit (che rappresntano RX1) collegati all'uscita del circuito di selezione dispositivo e delle linee dei dati. Notate che il bus di Figura 4-22 contiene il bus degli indirizzi e dei dati nonchè alcune o tutte le linee del bus di controllo. Questo "super" bus viene chiamato di solito *bus di*

sistema. Nel Capitolo 5, descriveremo il bus di sistema del Nanocomputer.
Riassumendo, la tecnica di base impiegata per inviare in uscita, su di un dispositivo di uscita, i dati provenienti da un registro, è molto semplice:

a. Generate, per mezzo del software, un solo impulso di selezione dispositivo di uscita. Le due istruzioni generalmente usate a questo scopo sono OUT (n),A e OUT (C),r in cui r è uno qualunque dei registri non specializzati a otto bit, A, B, C, D, E, H o L.
b. Usate l'impulso di selezione dispositivo per abilitare un latch nell'istante in cui i dati del registro appaiono sul bus bidirezionale dei dati.

La CPU dello Z80 è responsabile di tutto il processo di sincronizzazione. I latch nel dispositivo di uscita, giocano un ruolo passivo nel processo di trasferimento dei dati ed agiscono sui dati solo quando sono comandati da un impulso di selezione dispositivo. A seconda del tipo di dispositivo di latch usato, per agire sui dati viene usato un impulso di selezione dispositivo positivo o negativo.

ALCUNI CIRCUITI DI LATCH DI USCITA

Le Figure 4-22 e 4-23 mostrano alcuni circuiti di latch di uscita comunemente usati. Allo scopo di semplificare i disegni e mettere invece in risalto il modo in cui i dispositivi stessi sono collegati, la Figura 4-23 non mostra i circuiti che servono a generare gli impulsi di selezione dispositivo che forniscono gli impulsi di strobe ai circuiti di latch. Vi rimandiamo, invece agli schemi precedenti riguardanti la generazione di impulsi di selezione dispositivo, che vi mostrano in Figura 4-23 dove vengono effettuati i collegamenti dei fili provenienti dai circuiti degli impulsi di selezione.

L'INGRESSO NEI MICROCOMPUTER

Soffermiamoci ancora sul bus di Figura 4-21, con due trasmettitori, due ricevitori ed una CPU. Sono richieste due fasi affinchè TX1 invii tramite la CPU un singolo byte di dati ad RX1:

1. La CPU riceve i dati da TX1.
2. La CPU invia dati ad RX1.

Nella sezione relativa all'uscita del microcomputer, abbiamo discusso il Passo 2. In questa sezione relativa alla trattazione dell'ingresso nei microcomputer, discuteremo il Passo 1. La nostra supposizione è che TX1 sia un buffer/latch con un messaggio ad un byte caricato nei suoi flip-flop, e che le uscite Q di questi flip-flop, siano tramite, buffer disabilitati, logicamente scollegate dal bus. Osserviamo il circuito di Figura 4-24.

La combinazione buffer/latch permette al dispositivo trasmittente TX1 di conservare i dati senza porli sul bus. Se non ci fossero i buffer, le uscite Q dei flip-flop sarebbero collegate direttamente al bus, e quindi, in qualunque momento TX1 contenesse dei dati, questi verrebbero trasmessi anche sul bus. Ricordate che, su di un bus, è possibile effettuare una sola comunicazione di dati alla volta. Se TX1 fosse solo un latch a 8 bit anzichè un buffer latch a 8 bit, ne risulterebbe che, in qualunque momento TX1 avesse dei dati da trasmettere, i dati verrebbero trasmessi sul bus, impedendo l'effettuarsi di qualunque altra comunicazione. Questo non è accettabile. In genere, i dispositivi trasmittenti trattengono i dati per periodi di tempo relativamente lunghi, mentre sul bus si verificano altre comunicazioni fra altri dispositivi. Abilitando e disabilitando i buffer, un dispositivo ricevente può limitare il tempo di permanenza sul bus dei dati provenienti dal trasmettitore, aspettando il momento in cui esso (il ricevitore) è pronto per prelevare e leggere i dati dal bus. Perciò, i

buffer presenti su di un dispositivo di trasmissione permettono al dispositivo stesso di mantenere per lungo tempo i dati da trasmettere, prima che la trasmissione abbia effettivamente luogo, ed inviando i dati in modo indipendente.

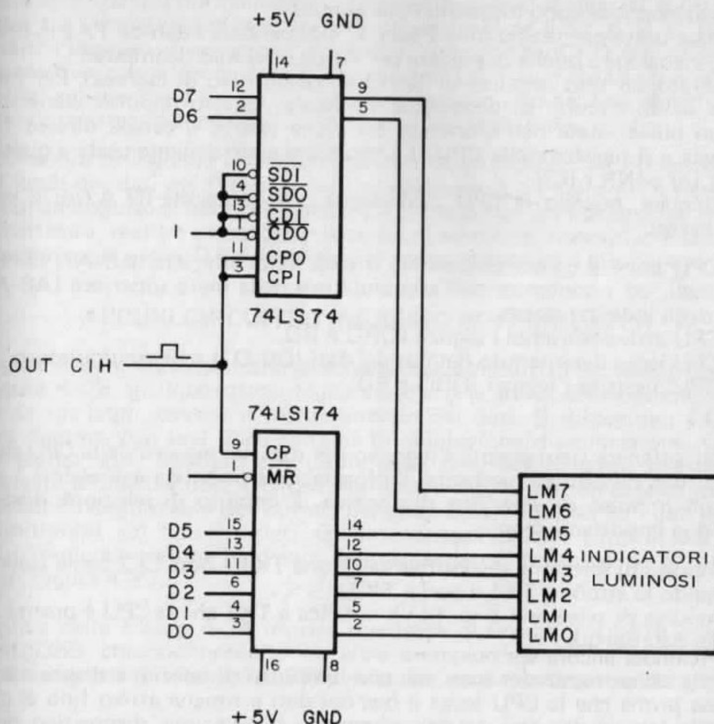


Figura 4-23. Circuito di latch per microcomputer basato sui latch 74LS174 e 74LS74.

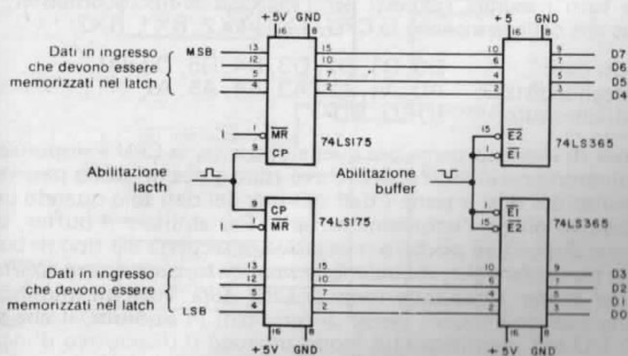


Figura 4-24. Circuito buffer/latch d'ingresso basato su due flip-flop 74LS175 quadrupli e su due Hex buffer 74LS365.

Ora dovrebbe essere chiaro perchè i dispositivi riceventi non hanno bisogno di buffer e di latch. Le uscite Q dei loro latch non sono collegate al bus, per cui quando un dispositivo ricevente è in possesso dei dati su cui è stato effettuato il latch, i dati non vengono nello stesso momento trasmessi sul bus. Da questo punto di vista, i dispositivi di ricezione sono dispositivi più semplici.

La tecnica usata per realizzare il Passo 1, cioè caricare i dati da TX1 in un registro della CPU, è analoga a quella impiegata per l'uscita del microcomputer.
Viene generato un solo impulso di selezione dispositivo di ingresso. Per mezzo del software e di un circuito di decodifica hardware. Questo impulso viene usato per abilitare un buffer-state nell'istante in cui viene aperto il canale diretto fra il bus bidirezionale e il registro della CPU. Le istruzioni generalmente usate a questo scopo sono IN A,(n) e INR r,(C).

In particolare, quando la CPU Z80 esegue una istruzione IN A,(n), si verificano i seguenti eventi:

1. La CPU pone il codice dispositivo n sulla metà inferiore (A0-A7) del bus degli indirizzi, ed i contenuti dell'accumulatore nella metà superiore (A8-A15) del bus degli indirizzi stesso.
2. La CPU attiva entrambi i segnali $\overline{IORQ}$ e $\overline{RD}$.
3. La CPU legge il contenuto del bus dei dati (D0-D7) nell'accumulatore.
4. La CPU disattiva i segnali $\overline{IORQ}$ e $\overline{RD}$.

Onde poter ottenere sicuramente l'ingresso dei dati nel registro della CPU può essere combinata, con decodifica hardware, l'informazione associata agli eventi 1 e 2 onde generare un impulso di selezione dispositivo. L'impulso di selezione dispositivo è usato per due importanti scopi:

1. L'impulso di selezione dispositivo seleziona TX1 e non TX2 come trasmettitore inviando lo strobe a TX1 e non a TX2.
2. L'impulso di selezione dispositivo notifica a TX1 che la CPU è pronta a leggere i suoi dati dal bus.

Dato che, le temporizzazioni sono tali che l'impulso di selezione dispositivo diventa attivo poco prima che la CPU legga il bus dei dati e rimane attivo fino al completamento della lettura dei dati dal bus, l'impulso di selezione dispositivo può essere usato per abilitare i buffer nel circuito di buffer/latch del trasmettitore TX1. In definitiva permette alle uscite degli otto flip-flop di essere direttamente collegate al bus dei dati.

Elenchiamo tutti i segnali richiesti per l'ingresso al microcomputer, che devono apparire sui bus che collega insieme la CPU, TX1, TX2, RX1, RX2:

Dal bus dei dati:	D0, D1, D2, D3, D4, D5, D6, D7
Dal bus degli indirizzi:	A0, A1, A2, A3, A4, A5, A7
Dal bus di controllo:	$\overline{IORQ}$, $\overline{RD}$

Per le operazioni di ingresso come per quelle di uscita, la CPU è responsabile di tutto il processo di sincronizzazione. Il buffer three-state gioca un ruolo passivo nel processo di trasferimento dei dati e pone i dati nel bus dei dati solo quando un impulso di selezione dispositivo gli dà l'istruzione di farlo. Per abilitare il buffer, si usa un impulso di selezione dispositivo positivo o negativo, a seconda del tipo di buffer usato.

Va notato in particolare che, *in qualunque momento, deve essere abilitato uno solo degli ingressi del buffer three-state verso la CPU Z80.* Tutti gli impulsi di selezione dispositivo d'ingresso dovrebbero essere decodificati in assoluto, il che significa che, per le porte di I/O per identificare in modo univoco il dispositivo d'ingresso desiderato, bisognerà usare tutti e otto i bit del codice dispositivo.
Se, a caricare i dati, viene chiamato un dispositivo non esistente, viene di solito caricato nel registro il byte FF.

Per riassumere, la tecnica base usata per effettuare l'ingresso di un dato in un registro della CPU Z80 è la seguente:

a. Generare, tramite il software, un singolo impulso di selezione dispositivo. Due sono le istruzioni normalmente interessate a questo scopo: IN A,(n) e IN r,(C), dove r è un qualunque registro non specializzato ad 8 bit, A, B, C, D, E, H o L.
b. Usare l'impulso di selezione dispositivo per abilitare i buffer three-state nel dispositivo trasmittente. Una volta abilitati i buffer, i dati da trasmettere sono inviati sul bus dei dati, da cui la CPU li legge in uno dei suoi registri interno. Il tipo di registro, dipende dall'istruzione di ingresso usata.

Nelle operazioni di ingresso ai microcomputer, quasi sempre la CPU gestisce direttamente il latch dei dati nei flip-flop del trasmettitore. La Figura 4-24 mostra un circuito in cui un impulso di selezione dispositivo è usato per il clock dei dati nei flip-flop del trasmettitore, mentre un secondo impulso di selezione dispositivo è usato per abilitare i buffer tra le uscite Q dei flip-flop del trasmettitore ed il bus dei dati.

ALCUNI CIRCUITI D'INGRESSO BUFFER/LATCH

Le Figure 4-24 e 4-25 mostrano gli schemi di due circuiti di ingresso di buffer/latch. Nella Figura 4-25, gli interruttori logici eseguono le stesse funzioni che verrebbero eseguite da un latch, ovvero il mantenimento dei dati. Il dispositivo 74LS365 ha sei buffer ognuno con una linea comune di abilitazione/disabilitazione. Gli ingressi di questi buffer sono collegati alle uscite degli interruttori logici, mentre le uscite sono collegate al bus dei dati, D0-D7. Per mezzo di un impulso di selezione dispositivo, si ottiene l'abilitazione dei buffer, ponendo così l'ingresso (o i dati del dispositivo trasmittente) sul bus dei dati. Gli interruttori logici e i buffer costituiscono insieme un'implementazione hardware del dispositivo di trasmissione TX1 del nostro esempio di Figura 4-21.

Il circuito della Figura 4-24 mostra due latch quadrupli 74LS175 con due Hex buffer 74LS365, che costituiscono un altro esempio di implementazione hardware di TX1. Gli ingressi dei latch 74LS175 possono essere collegati ad una qualunque

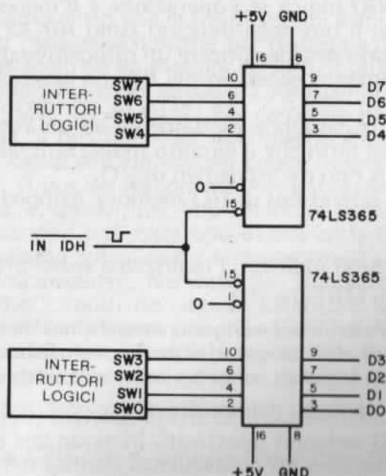


Figura 4-25. Un circuito d'ingresso basato su otto interruttori logici e sugli Hex buffer 74LS365.

sorgente di dati digitali, come, ad esempio, uno strumento di laboratorio. L'ingresso dei dati in un registro della CPU viene effettuato in due passi:

1. Effettuate un latch dei dati nei flip-flop 74LS175
2. Generate un impulso di selezione dispositivo per abilitare i buffer 74LS365, in modo che i dati vengono posti sul bus di dati e possono quindi essere letti nel registro.

Negli esperimenti, eseguirete il montaggio ed il controllo di circuiti simili a quelli presentati in questo capitolo.

TECNICHE DI I/O MEMORY MAPPED (I/O IN MAPPA DI MEMORIA)

Nel paragrafo precedente di questo capitolo abbiamo trattato le operazioni di I/O che vengono effettuate per mezzo di istruzioni appartenenti al gruppo di istruzioni di I/O, come per esempio IN A, (n), IN r, (C), OUT (n), A, OUT (C), r. Questo tipo di ingresso e di uscita del microcomputer viene chiamato I/O eseguite *tramite una porta*, perchè vi è un numero di porta di I/O a otto bit, specificato dal software, che dà inizio all'operazione. I segnali di sincronizzazione associati con l'I/O eseguito tramite una porta, sono IORQ e RD per l'ingresso, IORQ e WR per l'uscita.

Esiste un altro metodo per implementare L'I/O dei microcomputer: si tratta dell'I/O *memory mapped*, che sarà l'argomento di questo paragrafo. Le operazioni di I/O memory mapped in sostanza, considerano un dispositivo di I/O alla stessa stregua di una locazione di memoria. Quindi il dispositivo ha un indirizzo a sedici bit e si accede ad esso tramite le istruzioni che effettuano l'accesso in memoria, ad esempio le istruzioni LD. Per esempio, per porre in uscita un byte di informazioni proveniente dal registro B su di un dispositivo situato alla locazione C325, usando le tecniche di I/O memory mapped, si può usare l'istruzione LD (* C325H),B.

Analogamente, per inserire nel registro B dei dati provenienti da quel dispositivo, si potrebbe usare l'istruzione di ingresso memory mapped LD B,(C325H). L'I/O memory mapped e lo I/O tramite una porta, sono fondamentalmente, molso simili. In entrambi i casi, la CPU genera un segnale di richiesta ($\overline{MREQ}$ o $\overline{IORQ}$) ed un segnale di controllo ($\overline{RD}$ o $\overline{WR}$) indica se l'operazione è d'ingresso o d'uscita. Inoltre, sia in un caso che nell'altro, il bus degli indirizzi (solo A0-A7 per l'I/O tramite una porta) deve essere decodificato per identificare un dispositivo di I/O specifico. Infine, in entrambi i casi il trasferimento eccessivo dei dati ha luogo durante l'esecuzione di un ciclo macchina.

L'I/O memory mapped corrisponde esattamente ad un accesso ad una locazione di memoria, ad eccezione del fatto che il circuito indirizzato, anzichè essere un dispositivo di memoria, è un certo tipo di dispositivo di I/O.

Più specificatamente, le operazioni di I/O memory mapped richiedono i seguenti passi:

1. La CPU pone sul bus degli indirizzi, l'indirizzo a sedici bit del dispositivo.
2. La CPU attiva i segnali $\overline{MREQ}$ e $\overline{RD}$.
3. Le informazioni dei Passi 1 e 2 vengono inserite nel circuito di decodifica che produce un impulso di abilitazione dei buffer del dispositivo di ingresso, conseguentemente i dati in ingresso vengono posti sul bus dei dati.
4. La CPU legge i dati in ingresso dal bus dei dati.
5. I segnali MREQ e RD vengono disattivati, in modo che a sua volta venga disattivato l'impulso di selezione, di conseguenza i buffer nel dispositivo di ingresso e il dispositivo d'ingresso stesso ritorna nello stato di alta impedenza.
6. E' possibile riprendere altre comunicazioni sul bus.

L'uscita memory mapped viene realizzata attraverso una sequenza di passi molto simile alla precedente:

1. La CPU pone i dati in uscita sul bus dei dati e l'indirizzo a 16 bit, del dispositivo, sul bus indirizzi.
2. La CPU attiva i segnali $\overline{MREQ}$ e $\overline{WR}$.
3. Le informazioni dei Passi 1 e 2 vengono inserite nel circuito di decodifica che produce un impulso di attivazione per il clock dei flip-flop nel dispositivo di uscita, di conseguenza viene effettuata in uscita il latch dei dati.
4. La CPU disattiva i segnali $\overline{MREQ}$ e $\overline{WR}$ rimuove l'indirizzo del dispositivo dal bus degli indirizzi, e ritrasferisce i dati dal bus dei dati ponendo i buffer interni in uno stato di alta impedenza.
5. E' possibile riprendere altre comunicazioni sul bus.

L'uso dell'I/O memory mapped al posto dell'I/O tramite porta presenta pochissimi vantaggi. L'indirizzo a sedici bit, anche se rende complessi i circuiti di decodifica, espande lo spazio relativo all'indirizzo del dispositivo oltre i 256 disponibili con l'I/O tramite una porta. Per i dispositivi di uscita che presetano accumuli o conteggi, il processo di incremento può venire accelerato usando un'istruzione tipo INC (HL) al posto della sequenza di tre istruzioni IN A,(n), INC A, OUT (n), A che si renderebbero necessarie se si usasse L'I/O tramite una porta (supponendo che l'indirizzo sia già stato memorizzato in HL). Inoltre, se l'indirizzo si trova nella coppia di registri HL, le istruzioni del tipo LD r,(HL) e LD (HL),r sono più veloci delle loro corrispondenti istruzioni di I/O.

Il metodo usato per generare gli impulsi di sincronizzazione di memory mapped, è identico a quello impiegato per generare gli impulsi di sincronizzazione di accesso in memoria. In caso di I/O memory mapped, il metodo che si usa per collegare i buffer/latch d'ingresso e i latch di uscita è identico a quello usato nell'I/O tramite una porta. Pertanto, per gli schemi appropriati dei circuiti, vi rimanderemo alle discussioni fatte in precedenza su questi argomenti, nel Capitolo 3 ed in questo.

I BUS DEL MICROCOMPUTER

Nel Capitolo 2 abbiamo detto che in molti sistemi a microcomputer i segnali della CPU vengono bufferati realizzano il buffer per mettere a disposizione dell'utente una maggior potenza di pilotaggio ed un maggior fattore di protezione. Oltre alla presenza dei buffer, i segnali della CPU sono spesso tra loro combinati per generare nuovi segnali come $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IN}$ ed $\overline{OUT}$. Per certi bus standard, come il bus S-100, il Multibus® della Intel, il bus IEE 488, non specificatamente progettati per le CPU Z80, è necessaria della logica addizionale per creare i segnali corrispondenti alle specifiche dei bus standard. Il bus del Nanocomputer è stato definito per supportare delle strutture CPU multiple, e, quindi, per implementare l'accesso di una CPU esterna ad un altro bus di indirizzo dati e di controllo di una certa CPU occorrono determinati segnali. Quello che vogliamo sottolineare è che *raramente i segnali della CPU appaiono sul bus di un microcomputer.* Nel caso del Nanocomputer, che usa un nuovo standard il "Gamma Bus", molti dei segnali della CPU sono stati sottoposti a gate attraverso un singolo buffer. In questo caso i segnali della CPU sono riportati sul bus, al più ritardati di circa 30 nanosecondi. I segnali del bus provenienti dalla CPU e sottoposti a gate attraverso un singolo buffer, (che in molti casi è abilitato), sono

BA0, BA15, $\overline{BWR}$, $\overline{BRD}$, $\overline{BIORQ}$, $\overline{BMREQ}$, $\overline{BBUSAK}$, $\overline{BHALT}$, $\overline{BMI}$, $\overline{BFRSH}$ e BΦ

I segnali di ingresso alla CPU che si considerano bufferizzati prima del collegamento al bus sono:

$\overline{BWAIT}$, $\overline{BINT}$, $\overline{BNMI}$, $\overline{BBUSRQ}$ e $\overline{BRESET}$

Tutti questi segnali di ingresso sono portati a +5V attraverso un resistore di 930 ohm. Le linee dei dati BD0-BD7 non sono riportate nella precedente lista di segnali. I buffer che realizzano i gate di D0-D7 nel bus del Nanocomputer, sono abilitati dal

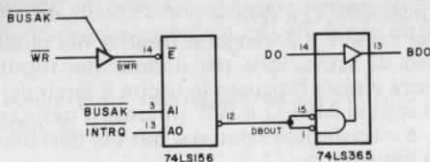


Figura 4-26. Circuito usato per abilitare l'uscita dei dati verso il bus del Nanocomputer.

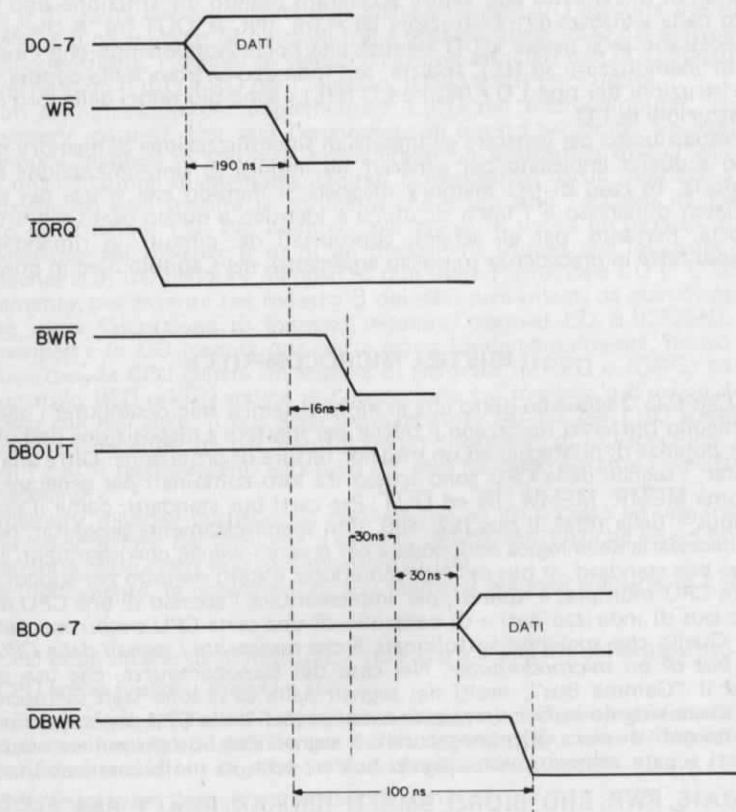


Figura 4-27. Diagramma delle temporizzazioni per una scrittura di I/O sul bus del Nanocomputer.

segnale $\overline{\text{DBOUT}}$ per l'uscita, e DBIN per l'ingresso. Ricordatevi che il bus dei dati è bidirezionale. La Figura 4-26 mostra come è generato il segnale $\overline{\text{DBOUT}}$ per abilitare i dati della CPU sul pin D0 verso la associata linea del Nanocomputer BD0.
Analizziamo la temporizzazione per una operazione di scrittura di I/O usando il caso peggiore di ritardo di propagazione di una porta del Low Power Schottky Data Book della SGS-ATES. Gli eventi sono i seguenti:

1. La CPU attiva $\overline{\text{IORQ}}$. Poco dopo, la CPU invia i dati su D0-D7.
2. La CPU attiva $\overline{\text{WR}}$. Non è importante l'esatto intervallo di tempo trascorso dopo l'attivazione di $\overline{\text{IORQ}}$.
3. Per crare il segnale $\overline{\text{BWR}}$, $\overline{\text{WR}}$ è sottoposto a gate attraverso un buffer abilitato da $\overline{\text{BUSAK}}$. Il ritardo è di 16 ns. (max).
4. $\overline{\text{BWR}}$ abilita il decodificatore 74LS156 per creare un segnale $\overline{\text{DBOUT}}$ attivo (sul basso). Il ritardo è di 30 ns. (max).
5. Il segnale $\overline{\text{DBOUT}}$ abilita un buffer 74LS365 che permette ai dati presenti sulla linea D0 di passare sulla linea BD0 del bus. Le altre linee dei dati sono collegate in modo simile al bus. Il ritardo è di 30 ns. (max).

Questi eventi sono riportati nel diagramma delle temporizzazioni di Figura 4-27.
Ora, supponiamo che dobbiate collegare un latch di uscita per il microcomputer usando $\overline{\text{WR}}$ e $\overline{\text{IORQ}}$ come clock del latch. La Figura 4-27 vi mostra chiaramente che quando $\overline{\text{WR}}$ diventa attivo i dati non saranno sul bus dei dati. Cosa comporterebbe l'uso di $\overline{\text{BWR}}$ al posto di $\overline{\text{WR}}$? Ancora, il dato non è presente. Il $\overline{\text{BWR}}$ deve essere ulteriormente ritardato per formare un segnale $\overline{\text{WR}}$ *ritardato da buffer*, chiamato $\overline{\text{DBWR}}$ e mostrato in Figura 4-27, con ritardo di 100 ns. rispetto a $\overline{\text{WR}}$. L'attivazione di $\overline{\text{DBWR}}$ e $\overline{\text{IORQ}}$ coincide con l'uscita dei dati validi sulle line BD0-BD7. Tutto questo spiega perchè è importante usare $\overline{\text{DBWR}}$ nei circuiti di uscita del Nanocomputer.

INTRODUZIONE AGLI ESPERIMENTI

Gli esperimenti che seguono illustrano le tecniche relative ai bus three-state e a collettore aperto, nonchè alcuni circuiti impiegati nell'I/O, tramite una porta, dei microcomputer.

Esperimento N.	Commenti
1	Illustra un bus con due diversi dispositivi di trasmissione ed uno di ricezione. Il bus è implementato con una linea ed un dispositivo di buffer quadruplo three-state 74LS125.
2	Illustra il comportamento di quattro invertitori 74LS05 i cui collettori aperti sono collegati ad un resistore di pull-up di 1000 ohm connessi a +5V.
3	Illustra un circuito di buffer/latch ad un bit usando un flip-flop D 74LS74 ed un buffer 74LS125 con "abilitazione attiva bassa".
4	Illustra un circuito di ingresso per un microcomputer ad 8 bit realizzato utilizzando due latch quadrupli 74LS175 e due hex buffer 74LS365.
5	Illustra l'I/O di un microcomputer con un contatore latch 74LS90 ed un circuito di buffer 74LS365 per l'ingresso ed un latch a quattro bit 74LS175 per l'uscita.

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è illustrare l'operatività di un buffer 74LS125 in un circuito a bus con due dispositivi di trasmissione ed un dispositivo di ricezione. I dispositivi di trasmissione sono implementati con interruttori logici sulla stazione di breadboarding del Nanocomputer. Il dispositivo di ricezione è rappresentato da un indicatore luminoso o lampadina (LED).

Passo 1

Montate il circuito dello schema precedente, il chip 74LS125 contiene quattro buffer, indipendenti, del bus. Ne userete soltanto due, uno per ogni dispositivo di trasmissione.

Passo 2

Ponete gli interruttori logici SW7 e SW1 nella posizione di livello logico 1, alimentate il breadboard. Spostate per parecchie volte l'interruttore logico SW0 da 1 logico a 0 logico. Che cosa osservate?

L'indicatore luminoso, inizialmente spento, rimane tale. Lo spostamento non ha nessun effetto.

Passo 3

Spostate SW7 nella posizione di 0 logico e muovete parecchie volte l'interruttore SW6. Che cosa osservate ora?

La luce dell'indicatore luminoso dovrebbe apparire tremolante. In tal modo avete abilitato il buffer three-state che collega l'interruttore logico SW6 al bus. Perciò il dispositivo di trasmissione SW6 e il dispositivo di ricezione LM0 stanno comunicando tramite il bus. Il dispositivo trasmittente continuerà a "rimanere in possesso" del bus finchè non verrà nuovamente isolato dal bus. Come si fa ad isolare dal bus il dispositivo di trasmissione? Basta disabilitare il buffer portando il suo pin di abilitazione a livello logico 1. Lasciando SW7 in posizione 0 logico ed SW1 ad 1 logico, spostate SW0 per parecchie volte. Che cosa osservate?

Non avreste dovuto riscontrare nessun effetto sul LM0 perchè il trasmettitore SW0 è isolato dal bus.

Passo 4

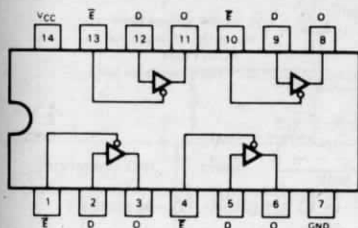
SENZA METTERLO IN PRATICA, provate a pensare che cosa succederebbe se sia SW1 che SW7 venissero posti simultaneamente al livello logico 0?

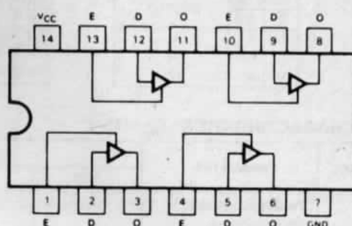
Entrambi i buffer verrebbero attivati, il che porterebbe ad una situazione *quanto mai indesiderabile.* Innanzitutto la comunicazione sul bus risulterebbe alterata perchè i due dispositivi cercherebbero di inviare contemporaneamente i dati sul bus stesso.

Configurazione dei pin del circuito integrato (Tavola 4-1)

Tavola 4-1. Caratteristiche del 74LS125.

QUAD 3-STATE BUFFERS WITH ACTIVE HIGH ENABLES





GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS125X T54LS126X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS125X T74LS126X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.4	3.4		V	I_{OH} = −1.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74	2.4	3.1		V	I_{OH} = −2.6 mA	V_{IL} per Truth Table
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0.35	0.5	V	I_{OL} = 24 mA	V_{IL} per Truth Table
I_{OZH}	Output Off Current HIGH				20	μA	V_{CC} = MAX, V_{OUT} = 2.4 V, V_E = V_{IL}	
I_{OZL}	Output Off Current LOW				−20	μA	V_{CC} = MAX, V_{OUT} = 0.4 V, V_E = V_{IL}	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−30		−130	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current, Outputs LOW	LS125			16	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 0 V	
		LS126			20	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 4.5 V	
	Power Supply Current, Outputs Off	LS125			20	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 4.5 V	
		LS126			24	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 0 V	

NOTES:

1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tavola 4-1. Caratteristiche del 74LS125 (seguito).

TRUTH TABLES

LS125

INPUTS		OUTPUT
$\bar{E}$	D	
L	L	L
L	H	H
H	X	(Z)

LS126

INPUTS		OUTPUT
E	D	
H	L	L
H	H	H
L	X	(Z)

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
(Z) = High Impedance (off)

AC CHARACTERISTICS: $T_A = 25°C$

SYMBOL	PARAMETER	LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
t_{PLH} t_{PHL}	Propagation Delay, Data to Output			10 16	ns	Fig. 2	$V_{CC} = 5.0$ V $C_L = 45$ pF $R_L = 667\ \Omega$
t_{PZH}	Output Enable Time to HIGH Level			16	ns	Figs. 4, 5	
t_{PZL}	Output Enable Time to LOW Level			30	ns	Figs. 3, 5	
t_{PLZ}	Output Disable Time from LOW Level			15	ns	Figs. 3, 5	$V_{CC} = 5.0$ V $C_L = 5$ pF $R_L = 667\ \Omega$
t_{PHZ}	Output Disable Time from HIGH Level			23	ns	Figs. 4, 5	

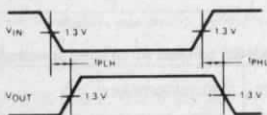


Fig. 1

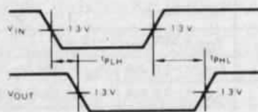


Fig. 2

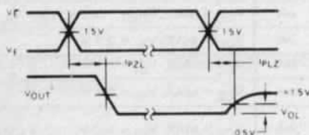


Fig. 3

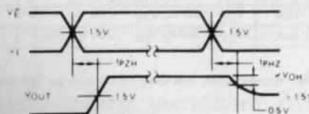
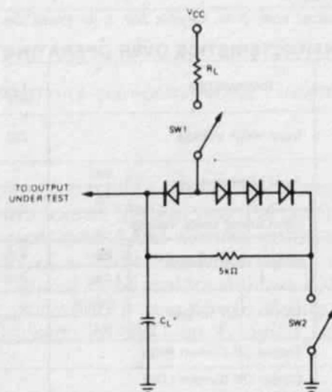


Fig. 4



SWITCH POSITIONS

SYMBOL	SW1	SW2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_{PLZ}	Closed	Closed
t_{PHZ}	Closed	Closed

Fig. 5

Schema del circuito (Figura 4-28)

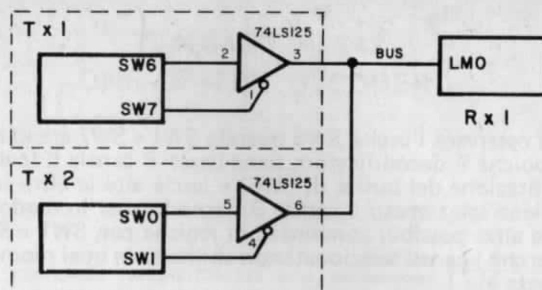


Figura 4-28. Schema N. 1: Un bus ad una linea con tre dispositivi.

In secondo luogo, il buffer potrebbe essere distrutto, non essendo progettato per operare in queste condizioni. Perciò è determinante il fatto che venga attivato un solo buffer alla volta.

Passo 5

Completate la seguente tabella:

SW1	SW2	Dati sul LMO
0	0	Disastro!
0	1	
1	0	
1	1	

Le vostre osservazioni dovrebbero annullare la validità della tabella seguente:

SW1	SW7	Dati sul LMO
0	0	Disastro!
0	0	SW0
1	0	SW6
1	1	Nessun dato

Passo 6

Prendete in considerazione la modifica apportata al circuito del bus illustrato in Figura 4-29. Il dispositivo 74LS42 è un decodificatore 1 a 10. La configurazione dei pin e la tabella della verità sono riportati nella Tavola 4-2.
Modificate il circuito del vostro bus per adattare il 74LS42 come detto sopra. Qual'è lo scopo del decodificatore 74LS42?

Lo scopo del 74LS42 è quello di abilitare un solo buffer per volta. Verificate che questo succeda compilando la tabella della pagina successiva.

SW1	SW7	Dati sul LM0
0	0	
0	1	
1	0	
1	1	

Avreste dovuto osservare l'uscita SW6 quando SW1 e SW7 erano entrambi nella posizione di OFF poichè il decodificatore tiene basso il canale 0 (collegato al pin di abilitazione/disabilitazione del buffer di SW6) e lascia alte le altre sue uscite. Con SW1 e SW7 bassi, viene selezionato il canale 3, facendo così in modo che LM0 rifletta i dati di SW0. Le altre possibili combinazioni logiche con SW1 e SW7, non portavano dati su LM0 perchè i canali selezionati non abilitano in quel momento nessun buffer. La tabella corretta è:

SW1	SW7	Dati sul LM0
0	0	SW6
0	1	Nessun dato
1	0	Nessun dato
1	1	SW6

In particolare non vi è nessuna combinazione logica di SW1 e SW7 che possa eventualmente nuocere ai componenti del circuito. Di conseguenza, questo secondo circuito viene preferito al precedente.

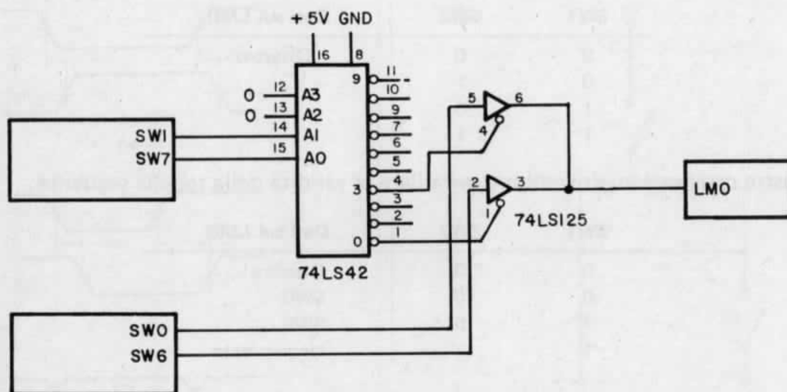


Figura 4-29. Schema N. 2.

ESPERIMENTO N. 2

Scopo

Lo scopo di questo esperimento è mostrare il comportamento di quattro invertitori 74LS05, i cui collettori aperti sono collegati ad un resistore di pull-up da 1000 ohm connesso a +5 volt.

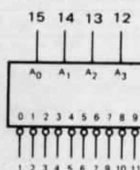
Tavola 4-2. Caratteristiche del 74LS42.

T54LS42/T74LS42
ONE-OF-TEN DECODER

DESCRIPTION — The LSTTL/MSI T54LS42/T74LS42 is a Multipurpose Decoder designed to accept four BCD inputs and provide ten mutually exclusive outputs. The LS42 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- **MULTI-FUNCTION CAPABILITY**
- **MUTUALLY EXCLUSIVE OUTPUTS**
- **DEMULTIPLEXING CAPABILITY**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**

LOGIC SYMBOL



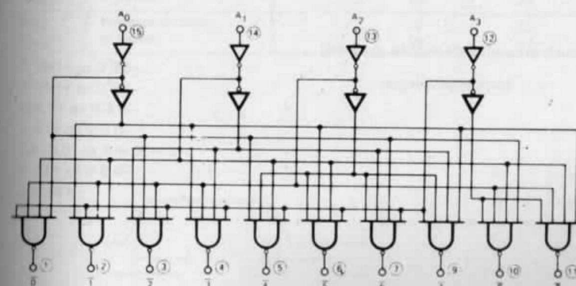
V_{CC} = Pin 16
GND = Pin 8

PIN NAMES

		LOADING (Note a) HIGH	LOW
$A_0 - A_3$	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{0}$ to $\overline{9}$	Outputs, Active LOW (Note b)	10 U.L.	5(2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



V_{CC} = Pin 16
GND = Pin 8
○ = Pin Numbers

CONNECTION DIAGRAM
DIP (TOP VIEW)

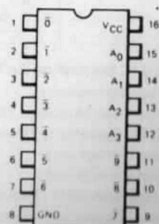


Tavola 4-2. Caratteristiche del 74LS42 (seguito).

FUNCTIONAL DESCRIPTION – The LS42 decoder accepts four active HIGH BCD inputs and provides ten mutually exclusive active LOW outputs, as shown by logic symbol or diagram. The active LOW outputs facilitate addressing other MSI units with active LOW input enables.

The logic design of the LS42 ensures that all outputs are HIGH when binary codes greater than nine are applied to the inputs.

The most significant input A_3 produces a useful inhibit function when the LS42 is used as a one-of-eight decoder. The A_3 input can also be used as the Data input in an 8-output demultiplexer application.

TRUTH TABLE

A_0	A_1	A_2	A_3	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$	$\overline{8}$	$\overline{9}$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
H	L	L	L	H	L	H	H	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H	H	H
H	H	L	L	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	H	H	H	L	H	H	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
H	H	H	L	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
L	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = HIGH Voltage Level
L = LOW Voltage Level

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	–65°C to +150°C
Temperature (Ambient) Under Bias	–55°C to +125°C
V_{CC} Pin Potential to Ground Pin	–0.5 V to +7.0 V
*Input Voltage (dc)	–0.5 V to +15 V
*Input Current (dc)	–30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	–0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs.

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
T54LS42X	4.5 V	5.0 V	5.5 V	–55°C to +125°C
T74LS42X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X package type: D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

Tavola 4-2. Caratteristiche del 74LS42 (seguito).

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Threshold Voltage for All Inputs
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA — V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0.35	0.5	V	I_{OL} = 8.0 mA — V_{IL} per Truth Table
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 4)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			7.0	12	mA	V_{CC} = MAX

NOTES:
1. Conditions for testing, not shown in the Table, are chosen to guarantee operation under "worst case" conditions.
2. The specified LIMITS represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature and supply voltage extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.
3. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
4. Not more than one output should be shorted at a time.

AC CHARACTERISTICS: T_A = 25°C

SYMBOL	PARAMETER	LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
t_{PLH} t_{PHL}	Propagation Delay (2 Levels)		11 18	18 25	ns	Fig. 2	V_{CC} = 5.0 V
t_{PLH} t_{PHL}	Propagation Delay (3 Levels)		12 19	20 27	ns	Fig. 1	C_L = 15 pF

AC WAVEFORMS

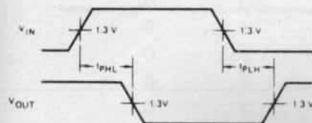


Fig. 1

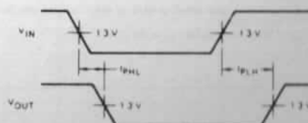
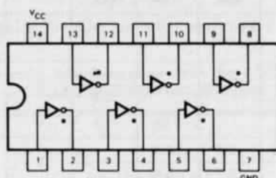


Fig. 2

Tavola 4-3. Caratteristiche del 74LS05.

HEX INVERTER



*OPEN COLLECTOR OUTPUTS

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS05X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS05X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
I_{OH}	Output HIGH Current				100	µA	V_{CC} = MIN, V_{OH} = 5.5 V, V_{IN} = V_{IL}
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{CCH}	Supply Current HIGH			1.2	2.4	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			3.6	6.6	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C **(See Page 273 for Waveforms)**

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output		14	22	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output		10	18	ns	C_L = 15 pF, R_L = 2.0 kΩ

NOTES

1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.

Passo 1

Montate il circuito come mostrato in Figura 4-30. Alimentate il bradboard e ponete i quattro interruttori logici a livello logico 0. L'indicatore luminoso LM0 è acceso o spento?

Noi abbiamo osservato che era acceso.

Schema del circuito (Figura 4-30)

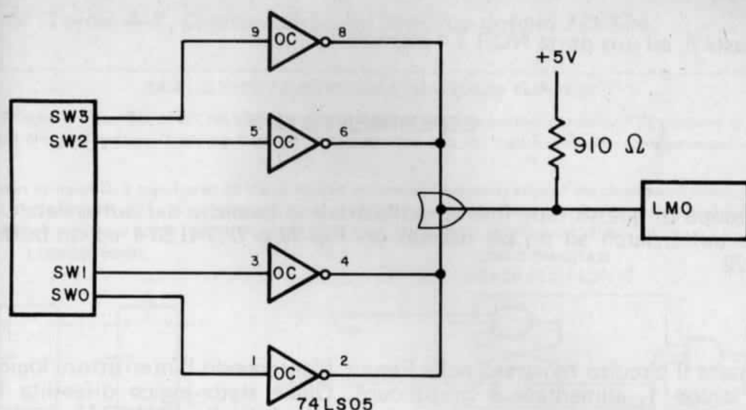


Figura 4-30. Schema N. 3.

Passo 2

Mantenendo SW1, SW2 e SW3 a livello logico 0, ponete SW0 a livello logico 1. Che cosa succede all'indicatore LM0?

Noi abbiamo osservato che l'indicatore LM0 rimane spento.

Passo 3

Convenendo che un LED acceso sia uguale al livello logico 1 ed un LED spento al livello logico 0, compilate la seguente tabella della verità:

SW0	SW1	SW2	SW3	LM0
0	0	0	0	0
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1.	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

A quale tipo di porta a quattro ingressi corrisponde la precedente tabella della verità?

La risposta è: ad una porta NOR a 4 ingressi.

ESPERIMENTO N. 3

Scopo

Lo scopo di questo esperimento è illustrare le tecniche del buffer/latch. Costruirete un buffer/latch ad un bit usando un flip-flop D 74LS74 ed un buffer driver 74LS125.

Passo 1

Montate il circuito mostrato nella Figura. Mantenendo l'interruttore logico SW0 a livello logico 1, alimentate il breadboard. Quale stato logico disabilita il buffer 74LS125? Quale trasmissione logica invia il clock al flip-flop 74LS74?

Lo stato logico 1 disabilita il buffer 74LS125, mentre la trasmissione logica 0-1 funge da clock per il flip-flop 74LS74.

Passo 2

Verificate sperimentalmente la seguente tabella della verità, relativa al circuito in oggetto:

SW0	SW1	P0	Uscita 74LS74 (LM0)	Uscita 74LS125 (LM1)
0	X	0	P	P = dati precedentemente sottoposti a latch
0	0	0-1-0 (⎍)	0	0
0	1	0-1-0 (⎍)	1	1
1	X	0	P	Stato di alta impedenza
1	0	0-1-0 (⎍)	0	Stato di alta impedenza
1	1	0-1-0 (⎍)	1	Stato di alta impedenza

Notate che l'uscita-Q segue l'ingresso-D solo quando il flip-flop riceve il clock. Quando non vi è una transizione da basso ad alto al suo ingresso di clock, CP, l'uscita-Q del flip-flop riflette i dati sottoposti a latch, che non deve essere necessariamente il valore logico dell'ingresso-D. Perciò l'uscita-Q, una volta settata, rimarrà costantemente indipendente dall'ingresso-D. Questa dimostra l'indipendenza di D da Q. Inoltre l'uscita del 74LS74, rappresentata da LM0 e l'uscita del 74LS175, rappresentata da LM1, sono indipendenti. LM1 seguirà l'uscita del 74LS74 solo quando il buffer è abilitato. Perciò, manipolando correttamente SW0, è possibile controllare ESATTAMENTE per quanto tempo l'uscita del flip-flop può restare connessa a LM1, o al bus.

Configurazioni dei pin dei circuiti integrati (Tavola 4-4 e 4-5).

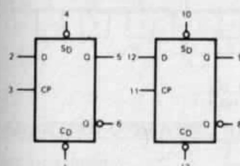
Tavola 4-4. Caratteristiche del flip-flop doppio 74LS74.

DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

DESCRIPTION – The T54LS74/T74LS74 dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and $\overline{Q}$ outputs.

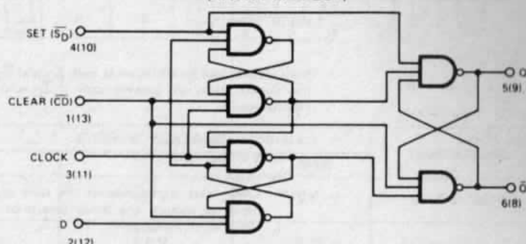
Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.

LOGIC SYMBOL



V_{CC} = Pin 14
GND = Pin 7

LOGIC DIAGRAM
(EACH FLIP-FLOP)



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE MIN	SUPPLY VOLTAGE TYP	SUPPLY VOLTAGE MAX	TEMPERATURE
T54LS74X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS74X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS (Note 1)
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current Data Clock, Set Clear				20 40 60	μA	V_{CC} = MAX, V_{IN} = 2.7 V
	Data Clock, Set Clear				0.1 0.2 0.3	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current Data Clock, Set Clear				−0.4 −0.8 −1.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			4.0	8.0	mA	V_{CC} = MAX, V_{CP} = 0 V

Tavola 4-4. Caratteristiche del flip-flop doppio 74LS74 (seguito).

MODE SELECT – TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{S}_D$	$\overline{C}_D$	D	Q	$\overline{Q}$
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

*Both outputs will be HIGH while both $\overline{S}_D$ and $\overline{C}_D$ are LOW, but the output states are unpredictable if $\overline{S}_D$ and $\overline{C}_D$ go HIGH simultaneously.

H,h = HIGH Voltage Level
L,l = LOW Voltage Level
X = Don't Care
l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

AC CHARACTERISTICS: $T_A = 25°C$ (See Page 274 for Waveforms)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
f_{MAX}	Maximum Clock Frequency		30	45		MHz	Fig. 1	
t_{PLH} t_{PHL}	Propagation Delay, Clock to Output			15 22	20 30	ns	Fig. 1	$V_{CC} = 5.0$ V, $C_L = 15$ pF
t_{PLH}	Propagation Delay,			10	15	ns	Fig. 2	
t_{PHL}	Set or Clear to Output	CP = L		18	24			
t_{PHL}		CP = H		26	35			

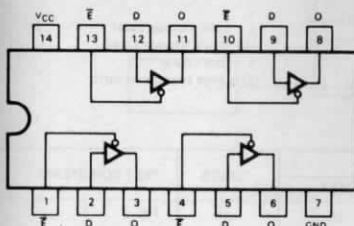
AC SET-UP REQUIREMENTS: $T_A = 25°C$ (See Page 274 for Waveforms)

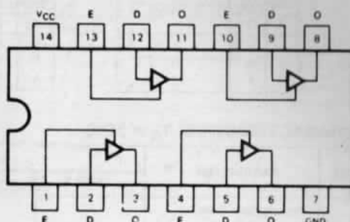
SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
$t_wCP(H)$	Clock Pulse Width (HIGH)	18	12		ns	Fig. 1	$V_{CC} = 5.0$ V
t_w	Set or Clear Pulse Width	15	10		ns	Fig. 2	
$t_s(H)$	Set-up Time HIGH, Data to Clock	10	6		ns	Fig. 1	
$t_h(H)$	Hold Time HIGH, Data to Clock	0	−14		ns		
$t_s(L)$	Set-up Time LOW, Data to Clock	20	14		ns		
$t_h(L)$	Hold Time LOW, Data to Clock	0	−6		ns		

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at $V_{CC} = 5.0$ V, $T_A = 25°C$.
3. Not more than one output should be shorted at a time.
4. SET-UP TIME (t_s) is defined as the minimum time required for the correct logic level to be present at the logic input prior to the clock transition from LOW to HIGH in order to be recognized and transferred to the outputs.
5. HOLD TIME (t_h) is defined as the minimum time following the clock transition from LOW to HIGH that the logic level must be maintained at the input in order to ensure continued recognition. A negative HOLD TIME indicates that the correct logic level may be released prior to the clock transition from LOW to HIGH and still be recognized.

Tavola 4-5. Caratteristiche del buffer quadruplo 74LS125.

QUAD 3-STATE BUFFERS WITH ACTIVE HIGH ENABLES





GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE MIN	SUPPLY VOLTAGE TYP	SUPPLY VOLTAGE MAX	TEMPERATURE
T54LS125X T54LS126X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS125X T74LS126X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.4	3.4		V	I_{OH} = −1.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74	2.4	3.1		V	I_{OH} = −2.6 mA	V_{IL} per Truth Table
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0.35	0.5	V	I_{OL} = 24 mA	V_{IL} per Truth Table
I_{OZH}	Output Off Current HIGH				20	μA	V_{CC} = MAX, V_{OUT} = 2.4 V, V_E = V_{IL}	
I_{OZL}	Output Off Current LOW				−20	μA	V_{CC} = MAX, V_{OUT} = 0.4 V, V_E = V_{IL}	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−30		−130	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current, Outputs LOW	LS125			16	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 0 V	
		LS126			20	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 4.5 V	
	Power Supply Current, Outputs Off	LS125			20	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 4.5 V	
		LS126			24	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 0 V	

NOTES:

1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tavola 4-5. Caratteristiche del buffer quadruplo 74LS125 (seguito).

TRUTH TABLES

LS125

INPUTS		OUTPUT
$\overline{E}$	D	
L	L	L
L	H	H
H	X	(Z)

LS126

INPUTS		OUTPUT
E	D	
H	L	L
H	H	H
L	X	(Z)

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
(Z) = High Impedance (off)

AC CHARACTERISTICS: $T_A = 25°C$

SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
t_{PLH} t_{PHL}	Propagation Delay, Data to Output			10 16	ns	Fig. 2	V_{CC} = 5.0 V C_L = 45 pF R_L = 667 Ω
t_{PZH}	Output Enable Time to HIGH Level			16	ns	Figs. 4, 5	
t_{PZL}	Output Enable Time to LOW Level			30	ns	Figs. 3, 5	
t_{PLZ}	Output Disable Time from LOW Level			15	ns	Figs. 3, 5	V_{CC} = 5.0 V C_L = 5 pF R_L = 667 Ω
t_{PHZ}	Output Disable Time from HIGH Level			23	ns	Figs. 4, 5	

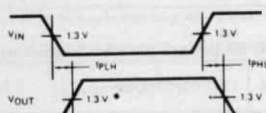


Fig. 1

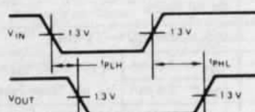


Fig. 2

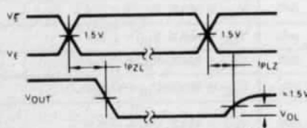


Fig. 3

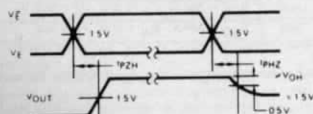
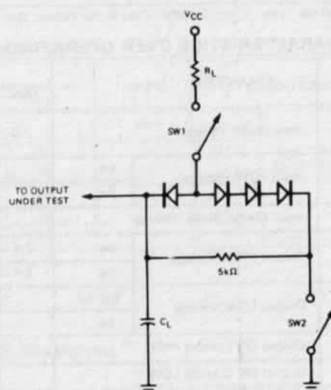


Fig. 4



SWITCH POSITIONS

SYMBOL	SW1	SW2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_{PLZ}	Closed	Closed
t_{PHZ}	Closed	Closed

Fig. 5

Schema del circuito (Figura 4-31).

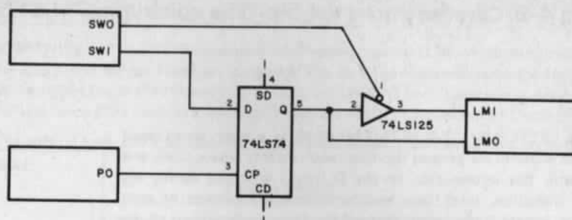


Figura 4-31. Schema N. 4.

Passo 3

Supponiamo che voi abbiate una tastiera con un bit pronto per essere letto dalla CPU. Non sapete quando la CPU leggerà il vostro bit, ma volete assicurarvi che al suo primo tentativo di lettura, il bit sarà pronto per essere letto. Che sequenza di passi effettuereste?

Una risposta è costituita da queste sequenze:

Passo 1. Posizionare SW1 per scegliere il bit che volete inviare alla CPU.

Passo 2. Spostare P0 per effettuare il latch del bit. L'uscita-Q del flip-flop 74LS74 è ora "agganciata" al vostro bit.

Passo 3. SW1 può ora essere modificato per selezionare il prossimo bit che volete inviare senza influenzare l'uscita-Q.

Quando è pronta per ricevere il bit d'ingresso, la CPU abiliterà il buffer, spostando, per mezzo della logica, SW0 da uno a zero a uno. Il segnale della CPU, per abilitare i buffer è temporizzato in modo da garantire alla CPU un tempo sufficiente per "afferrare" il bit (uguale all'uscita-Q) dal bus.

ESPERIMENTO N. 4

Scopo

Lo scopo di questo esperimento è quello di illustrare un circuito di ingresso di un microcomputer ad 8 bit utilizzando per il latch, due flip-flop quadrupli 74LS175, e per il buffer due hex buffer 74LS365.

Passo 1

Montate il circuito illustrato in Figura 4-32. In sostanza, questo circuito è solo un ampliamento del buffer/latch dell'Esperimento N. 3

a. E' un buffer/latch a otto bit anzichè un buffer/latch ad un bit.
b. L'interruttore di abilitazioni/disabilitazione del buffer SW0 è stato sostituito con l'uscita di un circuito di decodifica che genera l'impulso negativo di selezione dispositivo $\overline{\text{IN 12H}}$. $\overline{\text{IN 12H}}$ viene usato per fornire impulsi di strobe al pin E2, di abilitazione/disabilitazione, sul buffer 74LS365.

Configurazioni dei pin dei circuiti integrati (Tavole 4-6, 4-7, 4-8, 4-9).

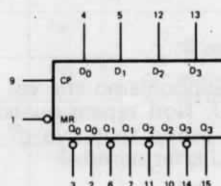
Tavola 4-6. Caratteristiche del flip-flop quadruplo 74LS125.

DESCRIPTION – The LSTTL/MSI T54LS175/T74LS175 is a high speed Quad D Flip-Flop. The device is useful for general flip-flop requirements where clock and clear inputs are common. The information on the D inputs is stored during the LOW to HIGH clock transition. Both true and complemented outputs of each flip-flop are provided. A Master Reset input resets all flip-flops, independent of the Clock or D inputs, when LOW.

The LS175 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- EDGE-TRIGGERED D-TYPE INPUTS
- BUFFERED-POSITIVE EDGE-TRIGGERED CLOCK
- ASYNCHRONOUS COMMON RESET
- TRUE AND COMPLEMENT OUTPUT
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

LOGIC SYMBOL



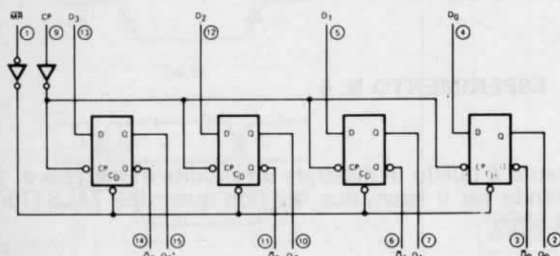
V_{CC} = Pin 16
GND = Pin 8

PIN NAMES

		LOADING (Note a)	
		HIGH	LOW
$D_0 - D_3$	Data Inputs	0.5 U.L.	0.25 U.L.
CP	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
$\overline{MR}$	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
$Q_0 - Q_3$	True Outputs (Note b)	10 U.L.	5(2.5) U.L.
$\bar{Q}_0 - \bar{Q}_3$	Complemented Outputs (Note b)	10 U.L.	5(2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



V_{CC} = Pin 16
GND = Pin 8
○ = Pin Numbers

CONNECTION DIAGRAM
DIP (TOP VIEW)

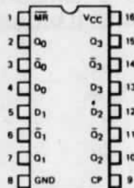


Tavola 4-6. Caratteristiche del flip-flop quadruplo 74LS175 (seguito).

FUNCTIONAL DESCRIPTION – The LS175 consists of four edge-triggered D flip-flops with individual D inputs and Q and $\overline{Q}$ outputs. The Clock and Master Reset are common. The four flip-flops will store the state of their individual D inputs on the LOW to HIGH Clock (CP) transition, causing individual Q and $\overline{Q}$ outputs to follow. A LOW input on the Master Reset ($\overline{MR}$) will force all Q outputs LOW and $\overline{Q}$ outputs HIGH independent of Clock or Data inputs.

The LS175 is useful for general logic applications where a common Master Reset and Clock are acceptable.

TRUTH TABLE

Inputs (t = n, $\overline{MR}$ = H)	Outputs (t = n+1) Note 1	
D	Q	$\overline{Q}$
L	L	H
H	H	L

Note 1: t = n + 1 indicates conditions after next clock.

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC} Pin Potential to Ground Pin	−0.5 V to +7.0 V
*Input Voltage (dc)	−0.5 V to +15 V
*Input Current (dc)	−30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	−0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs.

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
T54LS175X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
T74LS175X	4.75 V	5.0 V	5.25 V	0°C to +70°C

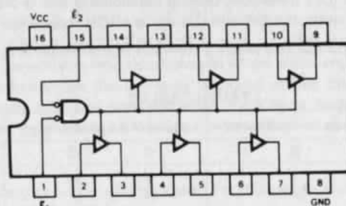
X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Threshold Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.4				
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 4)		−20		100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current			11	18	mA	V_{CC} = MAX	

Tavola 4-7. Caratteristiche del buffer esadecimale 74LS365.

T54LS365/T74LS365
HEX 3-STATE BUFFER WITH
COMMON 2-INPUT NOR ENABLE



TRUTH TABLE

INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	L
L	L	H	H
H	X	X	(Z)
X	H	X	(Z)

DESCRIPTION – The LS365/366/367/368 are high speed hex buffers with 3-state outputs. They are organized as single 6-bit or 2-bit/4-bit, with inverting or non-inverting data (D) paths. The outputs are designed to drive 15 TTL Unit Loads or 60 Low Power Schottky loads when the Enable ($\bar{E}$) is LOW.

When the Output Enable Input ($\bar{E}$) is HIGH, the outputs are forced to a high impedance "off" state. If the outputs of the 3-state devices are tied together, all but one device must be in the high impedance state to avoid high currents that would exceed the maximum ratings. Designers should ensure that Output Enable signals to 3-state devices whose outputs are tied together are designed so there is no overlap.

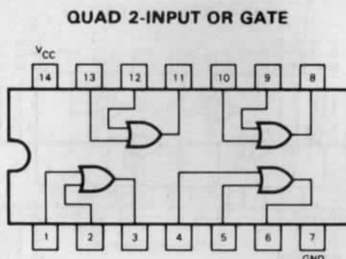
Programma UCINP

```
Codice
oggetto                Codice sorgente        Commenti

D3 11      UCINP:   OUT (11H),A         ; Sottoponi a latch i dati provenienti dagli
                                        ; interruttori logici
CD 9A 01            CALL WAIT           ; Ritarda per un po'
0E 12               LD C,12H            ; Setta il registro C con il codice del dispo-
                                        ; sitivo d'ingresso
ED 40               IN B,(C)            ; Abilitando il buffer carica i dati del latch
                                        ; nel registro
FF                  RST 038H            ; Restituisci il controllo al sistema operati-
                                        ; vo del Nanocomputer
210500     WAIT:    LD HL,0005H         ; Loop di ritardo
11FFFF     LOOP5:   LD DE,FFFFH
1B         LOOP6:   DEC DE
7A                  LD A,D
B3                  OR E
20FB                JR NZ,LOOP6
2B                  DEC HL
7D                  LD A,L
B4                  OR H
20F3                JR  NZ,LOOP5
C9                  RET
```

Tavola 4-8. Caratteristiche della porta 74LS32.

QUAD 2-INPUT OR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS32X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS32X	4.75 V	5.0 V	5.25 V	0° C to +70° C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IH}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = V_{IL}
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = V_{IL}
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			3.1	6.2	mA	V_{CC} = MAX, Inputs Open
I_{CCL}	Supply Current LOW			4.9	9.8	mA	V_{CC} = MAX, V_{IN} = 0 V

AC CHARACTERISTICS: T_A = 25°C (See Page 273 for Waveforms)

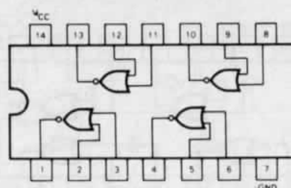
SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	7.0	11	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	7.0	11	ns	C_L = 15 pF

NOTES:

1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25° C.
3. Not more than one output should be shorted at a time.

Tavola 4-9. Caratteristiche della porta 74LS02.

QUAD 2-INPUT NOR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS02X	4.5 V	5.0 V	5.5 V	-55°C to 125°C
T74LS02X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type. D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			-0.65	-1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				-0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		-20		-100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.6	3.2	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			2.4	5.4	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 273 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

c. Il generatore di impulsi di clock P0, a flip-flop, è stato sostituito con un impulso positivo di selezione dispositivo, OUT 11H.
d. L'interruttore dell'ingresso-D è stato sostituito con gli otto interruttori logici, SW0-SW7, uno per ogni flip-flop.
e. L'indicatore luminoso LM1 è stato sostituito con le otto linee del bus dei dati, BD0-BD7, una per ogni bit che la CPU leggerà.
f. L'indicatore luminoso LM0 è stato sostituito con otto indicatori luminosi per indicare le uscite-Q sul latch a otto bit.

Schema del circuito (Figura 4-32).

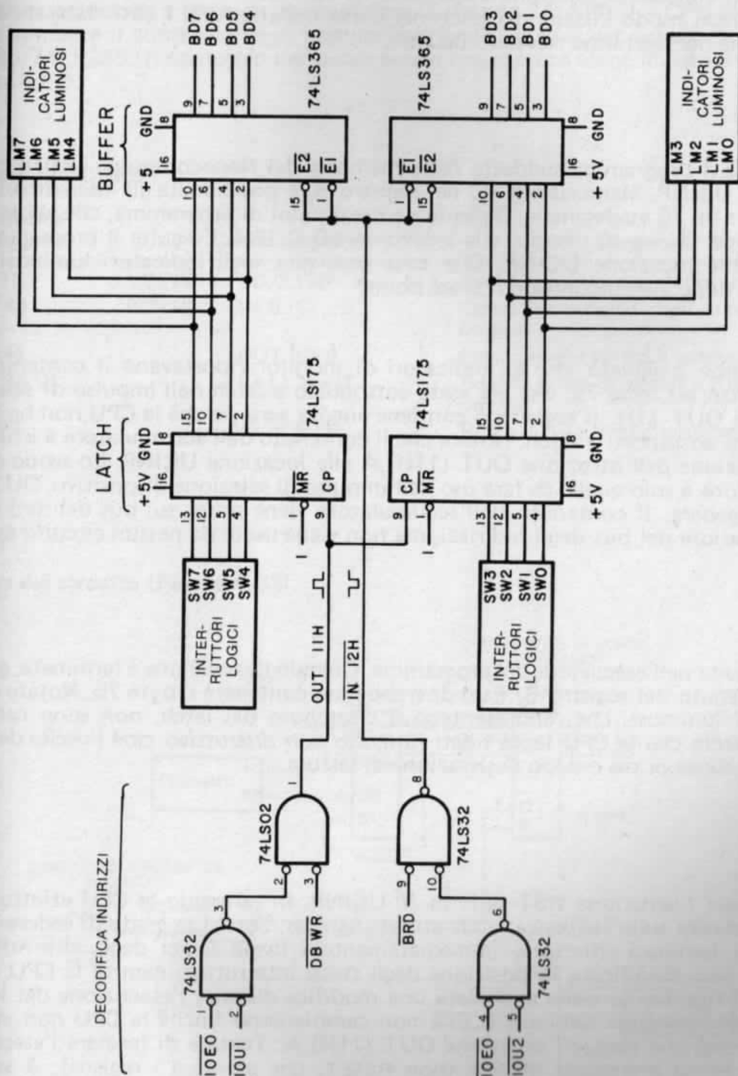


Figura 4-32. Schema N. 5.

La sorgente dei dati che vengono inseriti nella CPU è costituita dagli interruttori logici. Per prima cosa l'impulso di selezione dispositivo OUT 11H fornisce l'impulso di strobe ai dati contenuti nel latch a otto bit. Tali dati possono essere mantenuti per un tempo indefinito finchè un altro impulso OUT 11H non agisce come strobe per inviare i dati nei flip-flop o finchè la CPU non legge i dati abilitando gli otto buffer per mezzo dell'impulso di selezione dispositivo IN 12H. Naturalmente, la CPU non influenza in alcun modo l'uscita del latch nel corso della lettura. I dati resteranno dove sono finchè non verranno ricoperti da altri.

Passo 2

Caricate il programma suddetto nella memoria del Nanocomputer iniziando dalla locazione UCINP. Memorizzate 00 nel registro B, e posizionate gli interruttori logici per leggere un 75 esadecimale. Definite un breakpoint di programma, che abbia luogo subito dopo il loop di ritardo, alla locazione LD C,12H. Eseguite il programma iniziando dalla locazione UCINP. Che cosa osservate sugli indicatori luminosi e nel registro B dopo aver incontrato il breakpoint?

Noi abbiamo osservato che gli indicatori di monitor riportavano il contenuto del latch a otto bit, cioè 75, che era stato sottoposto a latch dall'impulso di selezione, dispositivi OUT 11H. Il registro B contiene ancora zero perchè la CPU non ha ancora letto i dati sottoposti a latch. Notate che il contenuto dell'accumulatore è irrilevante nell'esecuzione dell'istruzione OUT (11H),A alla locazione UCINP. Lo scopo di questa istruzione è solo quello di fare uso dell'impulso di selezione dispositivo, OUT 11H, che essa genera. Il contenuto dell'accumulatore viene posto sul bus dei dati e sulla metà superiore del bus degli indirizzi, ma non viene usato da nessun circuito esterno.

Passo 3

Proseguite nell'esecuzione del programma. Quando l'esecuzione è terminata, guardate il contenuto del registro B. Esso dovrebbe ora contenere il byte 75. Notate che gli indicatori luminosi, che rappresentano il contenuto del latch, non sono cambiati. Si dice perciò che la CPU legge i dati in modo *non distruttivo* cioè l'uscita del latch rimane la stessa prima e dopo l'operazione di lettura.

Passo 4

Cambiate l'istruzione RST 38H in JP UCINP. In tal modo la CPU effettuerà un loop indefinito sulla sequenza latch-attesa-ingresso. Sarete in grado di vedere che gli indicatori luminosi riflettono immediatamente i livelli logici degli interruttori, a meno di non modificare la posizione degli stessi interruttori mentre la CPU esegue il loop di ritardo. Se viene apportata una modifica durante l'esecuzione del loop di ritardo, gli indicatori luminosi (LED) non cambieranno finchè la CPU non effettua un altro loop che esegue l'istruzione OUT (11H),A. Tentate di fermare l'esecuzione del programma premendo BREAK (non RESET, che azzererà i registri). A seconda dell'istante in cui la CPU è azzerata, nella sequenza latch-attesa-ingresso, osserverete correlazioni diverse fra gli interruttori, gli indicatori luminosi ed il contenuto del registro B.

LASCIATE IN MEMORIA LA SUBROUTINE WAIT, SARA' USATA NELLO ESPERIMENTO SUCCESSIVO.

ESPERIMENTO N. 5

Scopo

Lo scopo di questo esperimento è illustrare un semplice circuito di I/O per microcomputer che controlla una sorgente di impulsi di clock. Gli impulsi di clock vengono contati e sottoposti a latch da un contatore a decade 74LS90. La CPU può leggere periodicamente il conteggio degli impulsi abilitando i buffer nel circuito buffer/latch 74LS90/74LS365. Il conteggio per poter essere visualizzato viene inviato in uscita sui quattro LED.

Programma UCINM

Codice oggetto		Codice sorgente	Commenti
0E 13	UCINM:	LD C,13H	; Poni 13 come codice dispositivo
ED 40	PCNTR:	IN B,(C)	; Carica il conteggio degli impulsi nel re- ; gistro B
ED 41		OUT (C),B	; Poni in uscita sui led il conteggio.
CD 9A 01		CALL WAIT	; Ritarda prima di leggere il conteggio suc- ; cessivo
18 F7		JR PCNTR	; Ripeti il ciclo lettura/scrittura/attesa

Schema del circuito (Figura 4-33).

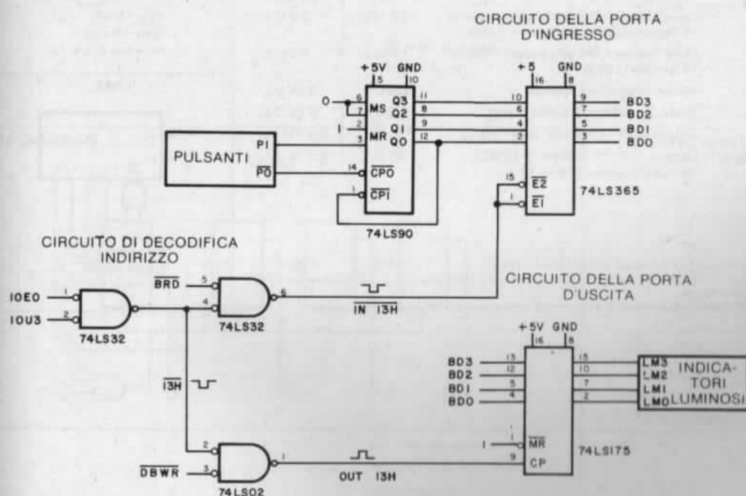


Figura 4-33. Schema N. 6.

Configurazioni dei pin dei circuiti integrati (Tavola 4-6 ÷ 4-11).

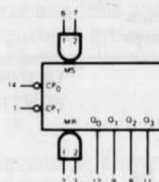
Tavola 4-10. Caratteristiche del contatore binario a 4 bit 74LS90.

DESCRIPTION – The T54LS90/T74LS90, T54LS92/T74LS92 and T54LS93/T74LS93 are high-speed 4-bit ripple type counters partitioned into two sections. Each counter has a divide-by-two section and either a divide-by-five (LS90), divide-by-six (LS92) or divide-by-eight (LS93) section which are triggered by a HIGH-tó-LOW transition on the clock inputs. Each section can be used separately or tied together (Q to $\overline{CP}$) to form BCD, bi-quinary, modulo-12, or modulo-16 counters. All of the counters have a 2-input gated Master Reset (Clear), and the LS90 also has a 2-input gated Master Set (Preset 9).

- **LOW POWER CONSUMPTION . . . TYPICALLY 45 mW**
- **HIGH COUNT RATES . . . TYPICALLY 50 MHz**
- **CHOICE OF COUNTING MODES . . . BCD, BI-QUINARY, DIVIDE-BY-TWELVE, BINARY**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**

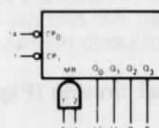
LOGIC SYMBOL

LS90



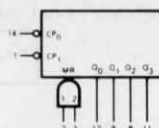
V_{CC} = Pin 5
GND = Pin 10
NC = Pins 4, 13

LS92



V_{CC} = Pin 5
GND = Pin 10
NC = Pins 2, 3, 4, 13

LS93



V_{CC} = Pin 5
GND = Pin 10
NC = Pins 4, 6, 7, 13

PIN NAMES		LOADING (Note a) HIGH	LOW
$\overline{CP}_0$	Clock (Active LOW going edge) Input to ÷2 Section	3.0 U.L.	1.5 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷5 Section (LS90), ÷6 Section (LS92)	2.0 U.L.	2.0 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷8 Section (LS93)	1.0 U.L.	1.0 U.L.
MR_1, MR_2	Master Reset (Clear) Inputs	0.5 U.L.	0.25 U.L.
MS_1, MS_2	Master Set (Preset-9, LS90) Inputs	0.5 U.L.	0.25 U.L.
Q_0	Output from ÷2 Section (Notes b & c)	10 U.L.	5(2.5) U.L.
Q_1, Q_2, Q_3	Outputs from ÷5 (LS90), ÷6 (LS92), ÷8 (LS93) Sections (Note b)	10 U.L.	5(2.5) U.L.

NOTES

a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges
c. The Q_0 Outputs are guaranteed to drive the full fan-out plus the $\overline{CP}_1$ input of the device.

Tavola 4-10. Caratteristiche del contatore binario a 4 bit 74LS90 (seguito).

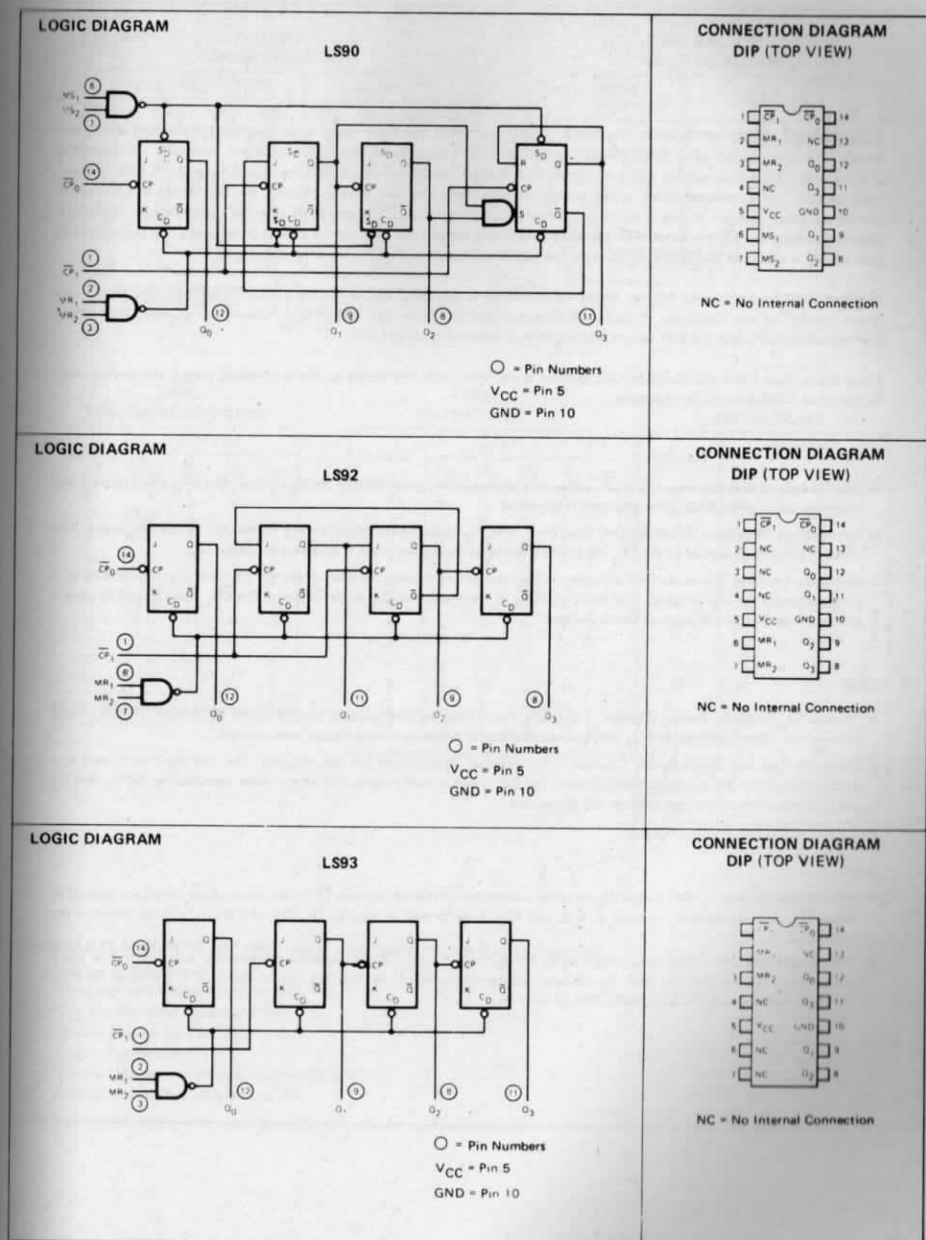


Tavola 4-10. Caratteristiche del contatore binario a 4 bit 74LS90 (seguito).

FUNCTIONAL DESCRIPTION – The LS90, LS92, and LS93 are 4-bit ripple type Decade, Divide-By-Twelve, and Binary Counters respectively. Each device consists of four master/slave flip-flops which are internally connected to provide a divide-by-two section and a divide-by-five (LS90), divide-by-six (LS92), or divide-by-eight (LS93) section. Each section has a separate clock input which initiates state changes of the counter on the HIGH-to-LOW clock transition. State changes of the Q outputs do not occur simultaneously because of internal ripple delays. Therefore, decoded output signals are subject to decoding spikes and should not be used for clocks or strobes. The Q_0 output of each device is designed and specified to drive the rated fan-out plus the $\overline{CP}_1$ input of the device.

A gated AND asynchronous Master Reset ($MR_1 \cdot MR_2$) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set ($MS_1 \cdot MS_2$) is provided on the LS90 which overrides the clocks and the MR inputs and sets the outputs to nine (HLLH).

Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.:

LS90

A. BCD Decade (8421) Counter – The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and a BCD count sequence is produced.

B. Symmetrical Bi-quinary Divide-By-Ten Counter – The Q_3 output must be externally connected to the $\overline{CP}_0$ input. The input count is then applied to the $\overline{CP}_1$ input and a divide-by-ten square wave is obtained at output Q_0.

C. Divide-By-Two and Divide-By-Five Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function ($\overline{CP}_0$ as the input and Q_0 as the output). The $\overline{CP}_1$ input is used to obtain binary divide-by-five operation at the Q_3 output.

LS92

A. Modulo 12, Divide-By-Twelve Counter – The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and Q_3 produces a symmetrical divide-by-twelve square wave output.

B. Divide-By-Two and Divide-By-Six Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function. The $\overline{CP}_1$ input is used to obtain divide-by-three operation at the Q_1 and Q_2 outputs and divide-by-six operation at the Q_3 output.

LS93

A. 4-Bit Ripple Counter – The output Q_0 must be externally connected to input $\overline{CP}_1$. The input count pulses are applied to input $\overline{CP}_0$. Simultaneous divisions of 2, 4, 8, and 16 are performed at the Q_0, Q_1, Q_2 and Q_3 outputs as shown in the truth table.

B. 3-Bit Ripple Counter – The input count pulses are applied to input $\overline{CP}_1$. Simultaneous frequency divisions of 2, 4, and 8 are available at the Q_1, Q_2, and Q_3 outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple-through counter.

Tavola 4-10. Caratteristiche del contatore binario a 4 bit 74LS90 (seguito).

LS90
MODE SELECTION

RESET/SET INPUTS				OUTPUTS			
MR_1	MR_2	MS_1	MS_2	Q_0	Q_1	Q_2	Q_3
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
L	X	L	X	Count			
X	L	X	L	Count			
L	X	X	L	Count			
X	L	L	X	Count			

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

LS92 AND LS93
MODE SELECTION

RESET INPUTS		OUTPUTS			
MR_1	MR_2	Q_0	Q_1	Q_2	Q_3
H	H	L	L	L	L
L	H	Count			
H	L	Count			
L	L	Count			

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

LS90
BCD COUNT SEQUENCE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H

NOTE: Output Q_0 is connected to Input CP_1 for BCD count

LS92
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	L	L	H
7	H	L	L	H
8	L	H	L	H
9	H	H	L	H
10	L	L	H	H
11	H	L	H	H

Note: Output Q_0 connected to input CP_1.

LS93
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

Note: Output Q_0 connected to input CP_1.

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	-65°C to +150°C
Temperature (Ambient) Under Bias	-55°C to +125°C
V_{CC} Pin Potential to Ground Pin	-0.5 V to +7.0 V
*Input Voltage (dc) for $\overline{CP}$	-0.5 V to +5.5 V
*Input Current (dc)	-30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	-0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or input Current limit is sufficient to protect the inputs

Tavola 4-10. Caratteristiche del contatore binario a 4 bit 74LS90 (seguito).

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC}) MIN	TYP	MAX	TEMPERATURE
T54LS90X T54LS92X T54LS93X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
T74LS90X T74LS92X T74LS93X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type. D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS (Note 1)
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current MS, MR $\overline{CP}_0$ $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				20 120 40 80	μA	V_{CC} = MAX, V_{IN} = 2.7 V
	MS, MR $\overline{CP}_0$, $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				0.1 0.4 0.8	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current MS, MR $\overline{CP}_0$ $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				−0.4 −2.4 −1.6 −3.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 4)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			9	15	mA	V_{CC} = MAX

NOTES

1. Conditions for testing, not shown in the table, are chosen to guarantee operation under "worst case" conditions.
2. The specified LIMITS represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature and supply voltage extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.
3. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C, and maximum loading.
4. Not more than one output should be shorted at a time.

Tavola 4-10. Caratteristiche del contatore binario a 4 bit 74LS90 (seguito).

AC CHARACTERISTICS: $T_A = 25°C$, $V_{CC} = 5.0$ V, $C_L = 15$ pF

SYMBOL	PARAMETER	LIMITS						UNITS	
		LS90		LS92		LS93			
		MIN	MAX	MIN	MAX	MIN	MAX		
f_{MAX}	$\overline{CP}_0$ Input Count Frequency	32		32		32		MHz	Fig. 1
f_{MAX}	$\overline{CP}_1$ Input Count Frequency	16		16		16		MHz	Fig. 1
t_{PLH} t_{PHL}	Propagation Delay, $\overline{CP}_0$ Input to Q_0 Output		16 18		16 18		16 18	ns	
t_{PLH} t_{PHL}	$\overline{CP}_1$ Input to Q_1 Output		16 21		16 21		16 21	ns	
t_{PLH} t_{PHL}	$\overline{CP}_1$ Input to Q_2 Output		32 35		16 21		32 35	ns	Fig. 1
t_{PLH} t_{PHL}	$\overline{CP}_1$ Input to Q_3 Output		32 35		32 35		51 51	ns	
t_{PLH} t_{PHL}	$\overline{CP}_0$ Input to Q_3 Output		48 50		48 50		70 70	ns	
t_{PLH}	MS Input to Q_0 and Q_3 Outputs		30					ns	Fig. 3
t_{PHL}	MS Input to Q_1 and Q_2 Outputs		40					ns	Fig. 2
t_{PHL}	MR Input to Any Output		40		40		40	ns	Fig. 2

AC SET-UP REQUIREMENTS: $T_A = 25°C$, $V_{CC} = 5.0$ V

SYMBOL	PARAMETER	LIMITS						UNITS	
		LS90		LS92		LS93			
		MIN	MAX	MIN	MAX	MIN	MAX		
t_W	$\overline{CP}_0$ Pulse Width	15		15		15		ns	Fig. 1
t_W	$\overline{CP}_1$ Pulse Width	30		30		30		ns	
t_W	MS Pulse Width	15						ns	Fig. 2, 3
t_W	MR Pulse Width	15		15		15		ns	Fig. 2
t_{rec}	Recovery Time MS to $\overline{CP}$	25						ns	Fig. 2, 3
t_{rec}	Recovery Time MR to $\overline{CP}$	25		25		25		ns	Fig. 2

RECOVERY TIME (t_{rec}) is defined as the minimum time required between the end of the reset pulse and the clock transition from HIGH-to-LOW in order to recognize and transfer HIGH data to the Q outputs.

AC WAVEFORMS

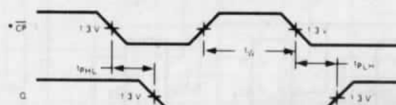


Fig. 1

*The number of Clock Pulses required between the t_{PHL} and t_{PLH} measurements can be determined from the appropriate Truth Tables.

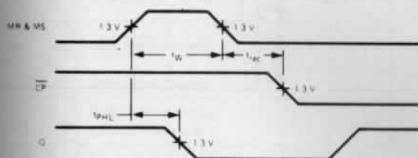


Fig. 2

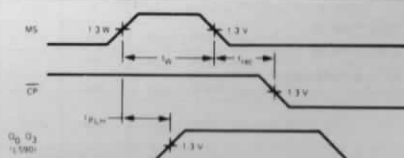


Fig. 3

Tavola 4-11. Caratteristiche del 74LS175.

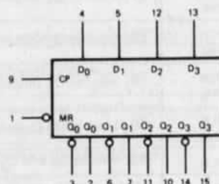
QUAD D FLIP-FLOP

DESCRIPTION – The LSTTL/MSI T54LS175/T74LS175 is a high speed Quad D Flip-Flop. The device is useful for general flip-flop requirements where clock and clear inputs are common. The information on the D inputs is stored during the LOW to HIGH clock transition. Both true and complemented outputs of each flip-flop are provided. A Master Reset input resets all flip-flops, independent of the Clock or D inputs, when LOW.

The LS175 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- **EDGE-TRIGGERED D-TYPE INPUTS**
- **BUFFERED-POSITIVE EDGE-TRIGGERED CLOCK**
- **ASYNCHRONOUS COMMON RESET**
- **TRUE AND COMPLEMENT OUTPUT**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**

LOGIC SYMBOL

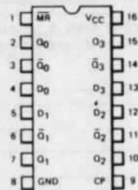


V_{CC} = Pin 16
GND = Pin 8

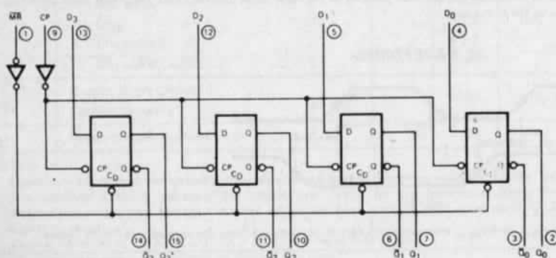
PIN NAMES		**LOADING** (Note a)	
		HIGH	LOW
$D_0 - D_3$	Data Inputs	0.5 U.L.	0.25 U.L.
CP	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
$\overline{MR}$	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
$Q_0 - Q_3$	True Outputs (Note b)	10 U.L.	5(2.5) U.L.
$\overline{Q}_0 - \overline{Q}_3$	Complemented Outputs (Note b)	10 U.L.	5(2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

CONNECTION DIAGRAM
DIP (TOP VIEW)



LOGIC DIAGRAM



V_{CC} = Pin 16
GND = Pin 8
○ = Pin Numbers

Tavola 4-11. Caratteristiche del 74LS175 (seguito).

FUNCTIONAL DESCRIPTION — The LS175 consists of four edge-triggered D flip-flops with individual D inputs and Q and $\overline{Q}$ outputs. The Clock and Master Reset are common. The four flip-flops will store the state of their individual D inputs on the LOW to HIGH Clock (CP) transition, causing individual Q and $\overline{Q}$ outputs to follow. A LOW input on the Master Reset ($\overline{MR}$) will force all Q outputs LOW and $\overline{Q}$ outputs HIGH independent of Clock or Data inputs.

The LS175 is useful for general logic applications where a common Master Reset and Clock are acceptable.

TRUTH TABLE

Inputs ($t = n$, $\overline{MR}$ = H)	Outputs ($t = n+1$) Note 1	
D	Q	$\overline{Q}$
L	L	H
H	H	L

Note 1: $t = n + 1$ indicates conditions after next clock.

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC} Pin Potential to Ground Pin	−0.5 V to +7.0 V
*Input Voltage (dc)	−0.5 V to +15 V
*Input Current (dc)	−30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	−0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs.

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC}) MIN	TYP	MAX	TEMPERATURE
T54LS175X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
T74LS175X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Threshold Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.4				
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 4)		−20		100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current			11	18	mA	V_{CC} = MAX	

Tavola 4-11. Caratteristiche del 74LS175 (seguito).

NOTES:
1. Conditions for testing, not shown in the Table, are chosen to guarantee operation under "worst case" conditions.
2. The specified LIMITS represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature and supply voltage extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.
3. Typical limits are at V_{CC} = 5.0 V, 25°C, and maximum loading.
4. Not more than one output should be shorted at a time.

AC CHARACTERISTICS: T_A = 25°C

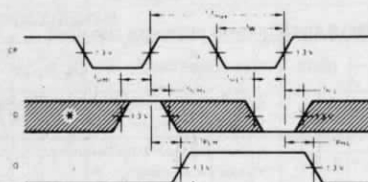
SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH} t_{PHL}	Propagation Delay, Clock to Output		12 15	20 22	ns	Fig. 1
t_{PHL}	Propagation Delay, $\overline{MR}$ to Output		20	28	ns	Fig. 2
f_{MAX}	Maximum Input Clock Frequency	30	45		MHz	Fig. 1

AC SET-UP REQUIREMENTS: T_A = 25°C

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
$t_W CP$	Minimum Clock Pulse Width	15	10		ns	Fig. 1
t_s	Set-up Time, Data to Clock (HIGH or LOW)	10			ns	Fig. 1
t_h	Hold Time, Data to Clock (HIGH or LOW)	0			ns	Fig. 1
t_{rec}	Recovery Time for $\overline{MR}$	12	8.0		ns	Fig. 2
$t_W \overline{MR}$	Minimum $\overline{MR}$ Pulse Width	12	8.0		ns	Fig. 2

AC WAVEFORMS

CLOCK TO OUTPUT DELAYS, CLOCK PULSE WIDTH, FREQUENCY, SET-UP AND HOLD TIMES DATA TO CLOCK



*The shaded areas indicate when the input is permitted to change for predictable output performance.

Fig. 1

MASTER RESET TO OUTPUT DELAY, MASTER RESET PULSE WIDTH, AND MASTER RESET RECOVERY TIME

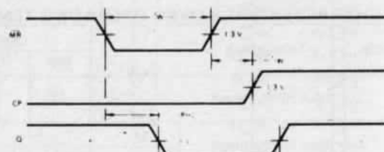


Fig. 2

DEFINITIONS OF TERMS:

SET-UP TIME (t_s) – is defined as the minimum time required for the correct logic level to be present at the logic input prior to the clock transition from LOW to HIGH in order to be recognized and transferred to the outputs.

HOLD TIME (t_h) – is defined as the minimum time following the clock transition from LOW to HIGH that the logic level must be maintained at the input in order to ensure continued recognition. A negative HOLD TIME indicates that the correct logic level may be released prior to the clock transition from LOW to HIGH and still be recognized.

RECOVERY TIME (t_{rec}) – is defined as the minimum time required between the end of the reset pulse and the clock transition from LOW to HIGH in order to recognize and transfer HIGH Data to the Q outputs.

Passo 1

Montate il circuito illustrato in Figura 4-33. Per azzerare i 4 flip-flop del 74LS90 spostate il generatore di impulsi P1. Lasciate P1 nella sua posizione di riposo in modo che l'ingresso di reset principale per il chip 74LS90 sia attivo (basso). Questo è necessario affinché il 74LS90 possa contare gli impulsi di clock. Caricate il programma suddetto ed eseguitelo. Che cosa osservate?

Noi abbiamo osservato che gli indicatori di monitor LM0-LM3 leggono zero. Nella esecuzione del programma, la CPU incontra prima l'istruzione IN B,(C), che determina la generazione di un impulso di selezione dispositivo, cioè IN 13H. L'impulso di selezione dispositivo $\overline{\text{IN 13H}}$ abilita i quattro buffer three-state 74LS365 permettendo alla CPU di "catturare" i dati del bus sui quattro pin di uscita del 74LS90. Dato che il 74LS90 è stato azzerato, e non ha ricevuto impulsi di clock al suo ingresso CP0, i 4 pin di uscita, che rappresentano il conteggio degli impulso avvenuti fino a quell'istante, registrano zero. Quindi la CPU trova zero nel registro B. L'istruzione successiva del programma è OUT (C),B. Viene generato un nuovo impulso di selezione dispositivo; questa volta il segnale OUT 13H, attivo alto. Esso viene usato per fornire impulsi di strobe al latch a quattro bit 74LS175 proprio nel momento in cui i dati del registro B si trovano sulle linee BD0-BD8 del bus dei dati. Dato che BD0-BD3 sono collegati agli ingressi D del flip-flop, 74LS175, il loro contenuto viene copiato sulle uscite Q quando ai flip-flop viene inviato l'impulso di strobe, OUT 13H. Gli indicatori luminosi LM0-LM3, essendo collegati alle uscite-Q, visualizzano il conteggio degli impulsi, che è zero. L'istruzione successiva è CALL di una subroutine che esegue un loop di ritardo. La CPU eseguirà tale routine prima di ritornare all'inizio e ripetere il processo precedente.

Passo 2

Mentre il programma è in esecuzione, spostare rapidamente il generatore di impulsi P0. Che cosa osservate?

Dovreste osservare che LM0-LM3 vengono periodicamente aggiornati. Dato che vi è un solo contatore a decade 74LS90, che può contare solo fino a 9, quello che state vedendo è l'*ultimo gate decimale* del *conteggio cumulativo degli impulsi* effettuato dal contatore.

Passo 3

Notate che, nel caso del circuito e del programma in oggetto, voi osservate l'esecuzione simultanea di due processi.

Il primo è un processo di conteggio nel quale gli impulsi provenienti da un generatore di impulsi vengono costantemente controllati da un circuito contatore/buffer latch. Il secondo processo è il processo di ingresso/uscita/attesa eseguito costantemente dalla CPU Z80. L'attesa potrebbe essere sostituita da un lavoro più significativo nel caso di una reale applicazione del microprocessore. Per esempio, la CPU potrebbe elaborare dei valori basati sulle letture del contatore. In particolare, se cambiasse il valore di conteggio degli impulsi, potrebbero essere calcolate la velocità e l'accelerazione della variazione del conteggio degli impulsi. Quello che è determinante, è che il conteggio è eseguito indipendentemente dalla CPU, seppure la CPU

non perde un colpo. La combinazione buffer/latch costituisce una tecnica importante al fine di assicurare questa indipendenza. La CPU può ''sbirciare'' il conteggio in qualunque momento ed assicurarsi della sua validità. Questo tipo di applicazione, utilizzante buffer e latch insieme, è divenuta molto comune.

Il prossimo capitolo tratta dell'hardware e del software utilizzati dal Nanocomputer per comandare la tastiera e i display a LED a sette segmenti. Una volta che avrete imparato ad usare l'hardware e il software del Nanocomputer per generare le vostre visualizzazioni, sarete meno interessati ad usare il tipo di visualizzazione descritto in quest'ultimo esperimento. Comunque, i semplici latch di uscita, come quello che avete appena montato, sono sempre utili e vale certamente la pena di conoscerli.

CAPITOLO 5

L'HARDWARE E IL SOFTWARE DI SISTEMA DEL NANOCOMPUTER

INTRODUZIONE

Negli ultimi capitoli ci siamo soprattutto interessati di *come* comunicare con un dispositivo Z80. L'opinione che a questo punto vi sarete formata è che la CPU, pur essendo determinante, non è altro che uno degli elementi di un sistema in cui un insieme di molteplici componenti deve interagire in modo armonico e sincronizzato. D'ora in avanti, se qualcuno cercherà di stupirvi citando statistiche che illustrano la vertiginosa discesa dei prezzi della CPU, voi sarete consapevoli del fatto che il costo della CPU è di gran lunga inferiore a quello dei mezzi necessari per comunicare con essa. In questo capitolo allarghiamo le nostre considerazioni sull'interfacciamento dello Z80 esaminando un caso reale, e precisamente il Nanocomputer della SGS-ATES. In special modo esamineremo dettagliatamente il microterminale tastiera/display del Nanocomputer, con particolare riferimento ai criteri seguiti per realizzarlo ed al software utilizzato per comunicare con esso. Al termine del capitolo sono descritti vari esperimenti da effettuare che vi illustreranno le modalità d'impiego della tastiera e del display. In tale modo saprete utilizzare, nel modo migliore, la tastiera/display nelle vostre applicazioni. Aspetti diversi, relativi all'hardware ed al sistema operativo del Nanocomputer, saranno trattati più sommariamente, lasciando a chi tra di voi possiede già il Nanocomputer, la cura di approfondire le sottigliezze del suo funzionamento interno.

OBIETTIVI

Al termine di questo capitolo sarete in grado di:

- Comprendere la struttura generale dell'hardware del Nanocomputer.
- Comprendere la struttura generale del software di sistema del Nanocomputer.
- Leggere e comprendere gli schemi relativi all'unità display del Nanocomputer.
- Localizzare nelle ROM le routine, appartenenti al sistema operativo del Nanocomputer, utilizzate per la gestione della unità display.
- Utilizzare le subroutine utilizzate per la gestione del display, appartenenti al sistema operativo del Nanocomputer, per realizzare delle visualizzazioni di vostra scelta.
- Leggere e comprendere gli schemi relativi alla tastiera del Nanocomputer.
- Localizzare nelle ROM le routine, appartenenti al sistema operativo del Nanocomputer, adibite a rilevare ed identificare univocamente i tasti premuti.
- Utilizzare le routine di ingresso dati dalla tastiera, appartenenti al sistema operativo del Nanocomputer, per definire relazioni, da voi scelte, tra la funzione della tastiera e i tasti.

- Leggere correttamente le routine di I/O seriale per il Nanocomputer, utilizzate in comunicazioni con periferiche seriali, quali registratori a cassette e terminali ASCII.

ASPETTI GENERALI DELL'HARDWARE DEL NANOCOMPUTER

La Figura 5-1 riproduce uno schema funzionale a blocchi relativo alla scheda base del Nanocomputer, mentre in Figura 5-2 appare il layout effettivo del circuito stampato. I componenti associati alle diverse funzioni sono contrassegnati ciascuno da un'etichetta. L'unità seriale ed il converter DC/DC, non compresi nel Nanocomputer standard, possono essere aggiunti facilmente, utilizzando il kit KNZ80 di espansione fornito dalla SGS-ATES.

Nei primi quattro capitoli del testo sono già state esaminate numerose funzioni del Nanocomputer. La CPU Z80 è stata trattata in modo particolareggiato nel corso del Capitolo 1. Nel Capitolo 3 sono state trattate la decodifica degli indirizzi di memoria sia per RAM che per EPROM/ROM, come pure la selezione delle porte di I/O relative alle varie periferiche del Nanocomputer. Il Capitolo 4 è stato dedicato alla definizione delle tecniche di gestione del bus e dell'I/O per i microcomputer. Gli argomenti ancora da trattare sono:

- Il bus del Nanocomputer, definito con il termine "bus di espansione", che comprende i driver del bus di espansione ed i connettori J1 e J2.
- Le unità a nastro magnetico e seriali ed i connettori J3a, J3b e J5 ad esse associati.
- Le 4 porte di I/O realizzate mediante due dispositivi PIO-Z80, ed i connettori J6 e J7 ad esse associate.

Dei primi due punti ci si occupera direttamente in questo capitolo, rimandando al Capitolo 7 la trattazione particolareggiata relativa al dispositivo di I/O, PIO.

Il bus del Nanocomputer

Le Figure 5-3 e 5-4 illustrano i circuiti utilizzati per la gestione del bus del Nanocomputer. I collegamenti elettrici che si possono stabilire con il bus del Nanocomputer sono classificabili in cinque categorie:

CATEGORIA 1: *Linee pilotate da Driver con Buffer Three-State.*

Qualunque linea collegata sul bus del Nanocomputer ad un segnale della categoria 1, è pilotata da un driver con buffer three-state, in grado di assorbire (verso massa) una corrente massima di 24mA con tensione fino a 0,5 volt, ed erogare (prelevandola dalla alimentazione a +5V) una corrente di 2,6mA per una tensione superiore a 2,4 volt. Questi driver passano nel loro stato ad alta impedenza quando $\overline{\text{BUSAK}}$ è al livello logico 1 è quindi importante fare in modo che le linee collegate ad un segnale del bus della categoria 1 non siano pilotate da più di un driver per volta.

CATEGORIA 2: *Linee Bidirezionali.*

I driver delle linee della categoria 2 sono simili a quelli della categoria 1. Mentre i ricevitori collegati a queste linee sono normalmente dei dispositivi 74LS caratterizzati nello stato basso (meno di 0,4V), da una "source current" di 0,36 mA, e nello stato alto

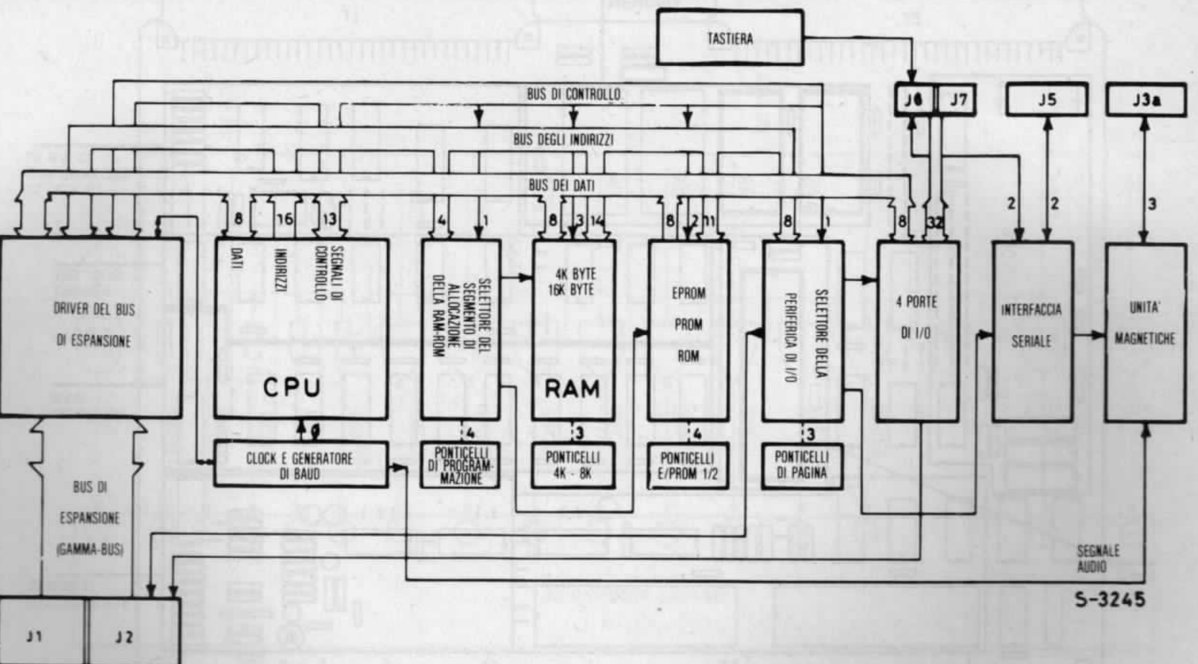


Figura 5-1. Schema a blocchi della scheda base del Nanocomputer.

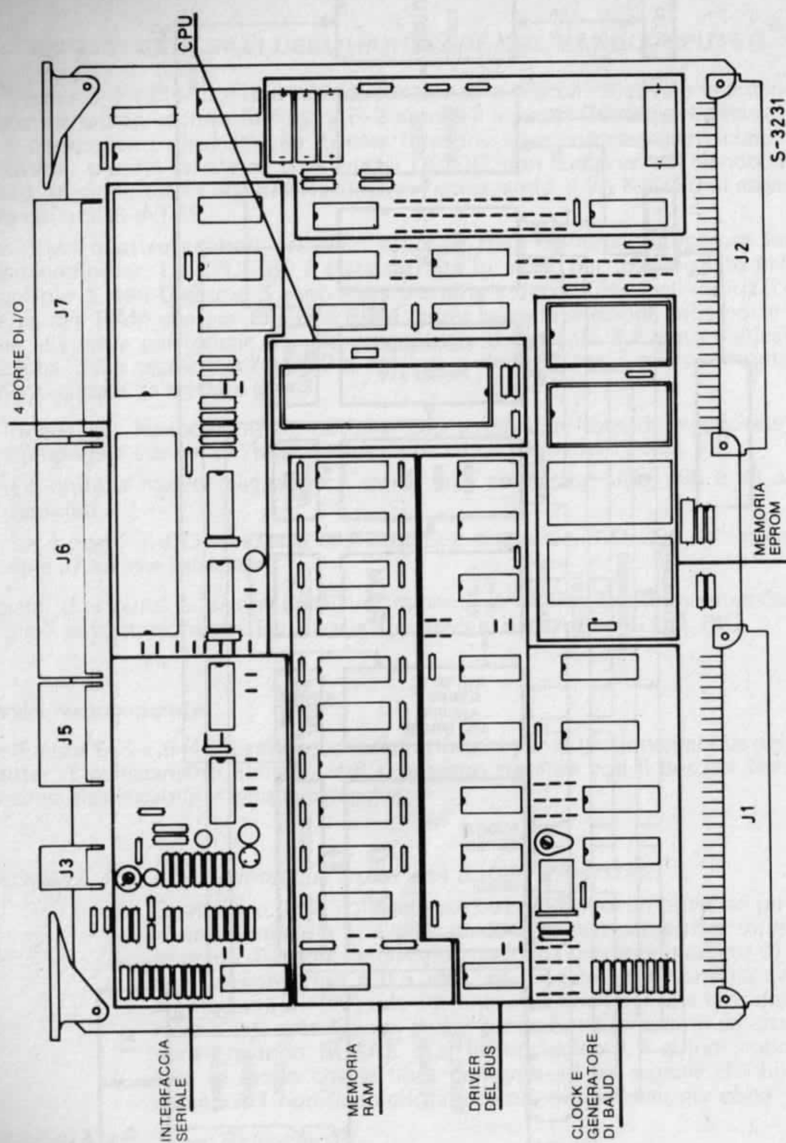


Figura 5-2. Layout del circuito stampato del Nanocomputer.

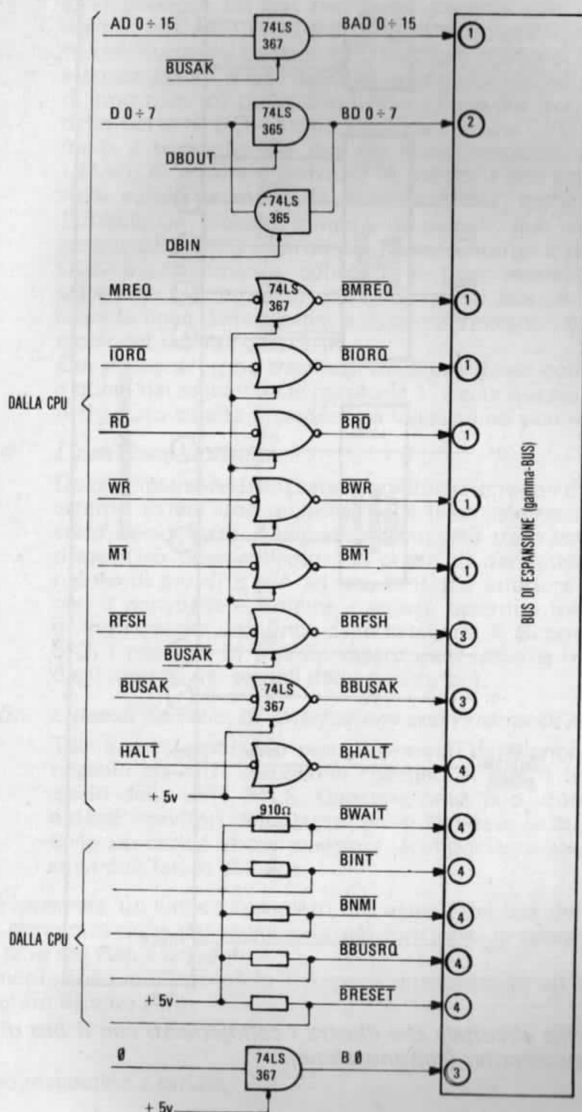


Figura 5-3. Schema circuitale che illustra i collegamenti con il bus di espansione del Nanocomputer.

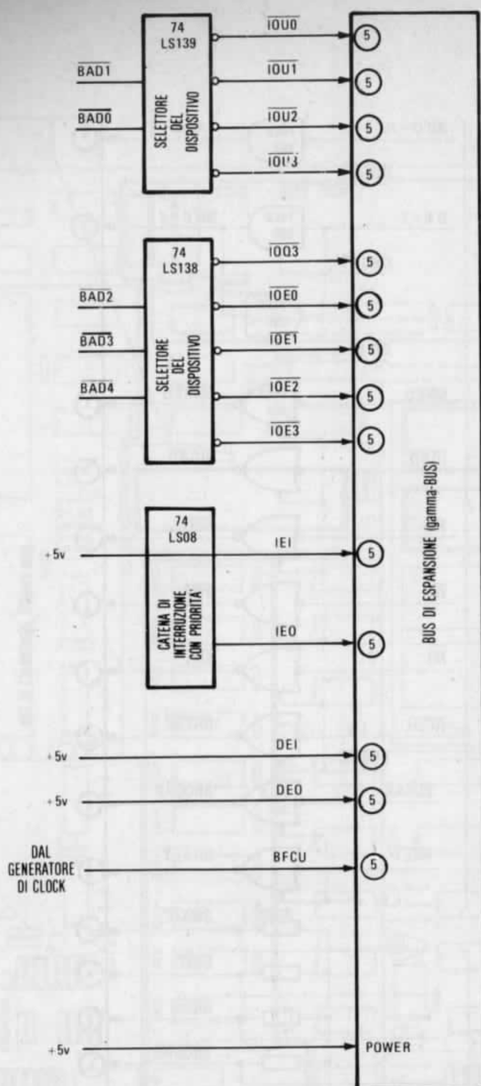


Figura 5-4. Schema circuitale che illustra i collegamenti con il bus di espansione del Nanocomputer (continuazione).

(più di 2,7V), da una capacità di assorbimento pari a 20 microampere. Le linee bidirezionali del bus del Nanocomputer sono controllate dai segnali $\overline{\text{DBIN}}$ e $\overline{\text{DBOUT}}$, che non possono mai essere contemporaneamente attivi.

CATEGORIA 3: *Linee che Devono essere Pilotate dalla CPU Z80 del Nanocomputer*

Alcuni segnali di controllo devono essere forniti solamente dalla CPU Z80 della scheda del Nanocomputer, e mai da qualche altra CPU presente sul bus del Nanocomputer. Un interessante argomento, che però esula dall'ambito di questa trattazione, è appunto costituito dai bus con CPU multiple residenti. Il Nanocomputer è concepito in modo tale che un'altra CPU Z80 (oppure una CPU di tipo diverso) possa condividere l'uso dei bus, della memoria e delle porte di I/O. Questo significa che una CPU esterna può assumere il controllo del bus del Nanocomputer, inviare segnali di I/O e/o di lettura e scrittura in memoria per accedere alle periferiche ed alla memoria del Nanocomputer, come se fossero le sue! Tuttavia un piccolo numero di segnali, può essere unicamente gestito dalla CPU interna del Nanocomputer e non da altri dispositivi eventualmente collegati al bus; essendo tali dispositivi solamente dei semplici utenti di questi bus. A questa categoria ossia le linee della categoria 3, corrispondono i soli segnali $\overline{RFSH}$, clock del sistema e $\overline{BUSAK}$.

Questi segnali sono trasmessi mediante driver con buffer, analoghi a quelli dei segnali della categoria 1, ma in questo caso il passaggio nello stato ad alta impedenza è soggetto ad alcune restrizioni.

CATEGORIA 4: *Linee Open-Collector.*

Queste linee sono utilizzate in combinazione con circuiti "wired-or" esterni, in cui una qualsiasi delle linee esterne che si porti nello stato basso forza il segnale sul bus allo stato basso. Un qualsiasi dispositivo open-collector, in grado di dar luogo ad un assorbimento di più di 8 mA ad una tensione inferiore a 0,5 volt, è idoneo a convogliare insieme i segnali esterni e formare un segnale di ingresso per una linea della categoria 4. Si notino, nella Figura 5-3, i resistori di pull-up raggruppati insieme in corrispondenza degli ingressi dei segnali della categoria 4.

CATEGORIA 5: *Linee di Servizio, di Interruzione con Priorità di Accesso.*

Tali linee trasportano segnali generati dal Nanocomputer ed utilizzabili da altri dispositivi collegati al bus. I livelli logici sono quelli della serie 74LS. Ciascuna linea può, quindi, servire sino a dieci ricevitori della stessa serie. Nel caso in cui si renda necessario un carico ancor maggiore, è importante scegliere dei dispositivi con fan-in più alto.

La Tabella 5-1 presenta un elenco completo dei segnali del bus del Nanocomputer con la relativa denominazione del segnale, la sua funzione, la categoria e il numero del pin di J1 e J2 in cui esso è presente

Per una documentazione più completa rivolgersi direttamente all'SGS-ATES, Via Olivetti, 2 di Agrate Brianza (Mi).

Le unità a nastro magnetico e seriale

Il Nanocomputer può collegare sino ad un massimo di due unità a cassette. Il connettore J3 (a e b) è adibito esclusivamente al collegamento delle due unità a cassette con il Nanocomputer. La sensibilità d'ingresso richiesta a ciascuna delle unità a cassetta è compresa tra 10 e 50 millivolt. Per le regolazioni occorre servirsi del potenziometro R54 posto sulla scheda del Nanocomputer, oppure di una rete di resistori esterni. L'uscita di ciascuna delle cassette deve essere pari a 300-400 milli-

Tabella 5-1. Il bus di espansione del Nanocomputer connettori J1 e J2.

Nome	Funzione	Cat.	Pin
+ 5V	Alimentazione di 5V ± 2%	N.A.	J1-1ac/J2-1ac
$\overline{BMREQ}$	Richiesta di Memoria	1	J2-22c
$\overline{BIORQ}$	Richiesta di Input-Output	1	J2-21c
$\overline{BRD}$	Lettura di dati dalla memoria o da Periferiche	1	J2-13c
$\overline{BWR}$	Scrittura in memoria o periferica	1	J2-14c
$\overline{BMI}$	Ciclo di Macchina 1	1	J1-15c
$\overline{BRFSH}$	Ciclo di Refresh	1	J1-11c
$\overline{BBUSAK}$	Riconoscimento del Bus	3	J1-20c
$\overline{BBUSRQ}$	Richiesta del Bus	4	J2-25c
$\overline{BHALT}$	Halt	3	J1-18c
$\overline{BWAIT}$	Attesa	4	J2-26c
$\overline{BINT}$	Richiesta di Interruzione	4	J2-24c
$\overline{BNMI}$	Richiesta di Interruzione non mascherabile	4	J2-23c
$\overline{BRESET}$	Reset	4	J1-28c
B0	Clock di Macchina	3	J1-17C
$\overline{BFCU}$	Clock di Conversione	5	J1-8c
$\overline{IOU0}$	Decodifica dell'Indirizzo per le linee BAD0, BADL	5	J1-24c
$\overline{IOU1}$		5	J1-23c
$\overline{IOU2}$		5	J1-22c
$\overline{IOU3}$		5	J1-21c
$\overline{IOQ3}$		5	J2-5c
$\overline{IOE0}$	Decodifica parziale dell'indirizzo (3, 4, 5, 6, 7) per le linee BAD2, BAD3, BAD4.	5	J2-6c
$\overline{IOE1}$		5	J2-8c
$\overline{IOE2}$		5	J2-9c
$\overline{IOE3}$		5	J2-11c
+ 12V		N.A.	J1-16ac/J2-16ac
— 12V			J2-3ac
— 5V			J2-4ac
GND	Ritorno della Linea di Alimentazione	N.A.	J1-32ac/J2-32ac
BD0	Bit meno significativo	2	J1-27c
BD1	Linee dei Dati	2	J1-26c
BD2		2	J1-25c
BD3		2	J1-29c
BD4		2	J1-30c
BD5		2	J1-31c
BD6		2	J2-12c
BD7	Bit più significativo	2	J2-10c
BA0	Bit meno significativo	1	J1-3c
BA1	Linee degli indirizzi	1	J1-7c
BA2		1	J1-6c
BA3		1	J1-3c
BA4		1	J1-4c
BA5		1	J1-5c
BA6		1	J2-30c
BA7		1	J2-29c

Tabella 5-1. Continua.

Nome	Funzione	Cat.	Pin
BA8		1	J2-31c
BA9		1	J2-28c
BA10		1	J1-12c
BA11	Linee degli indirizzi	1	J2-27c
BA12		1	J2-19c
BA13		1	J2-7c
BA14		1	J2-18c
BA15	Bit più significativo	1	J2-17c

Nota: Le linee del bus BA0-BA15 sono quelle stesse che, sulla piastra di sperimentazione, recano l'etichetta BA0-BA15.

Tabella 5-2. Connettori J3a e J3b.

Pin	Nome	Funzione
1	IM1	Ingresso audio della cassetta N. 1
2	IM2	Ingresso audio della cassetta N. 2
3	GND	
4	GND	
5	UM1	Uscita audio verso la cassetta N. 1
6	UM2	Uscita audio verso la cassetta N. 2
7	CA1ON	Uscita per start/stop automatico della cassetta N. 1
8	CA2ON	Uscita per start/stop automatico della cassetta N. 2

volt. Inoltre l'interfaccia per cassette controlla il meccanismo di scorrimento del nastro dell'unità tramite il pin *"REMOTE CONTROL"* (controllo a distanza). I particolari, relativi ai criteri di realizzazione dell'interfaccia sono, insieme alla descrizione delle routine LOAD e DUMP, trattati più avanti nel corso di questo stesso capitolo, essendo la comunicazione seriale realizzata mediante software. La Tabella 5-2 riporta una descrizione sommaria del connettore J3.

L'interfaccia seriale del Nanocomputer ammette modalità di comunicazione conforme ai seguenti criteri normalizzati di interfaccia: RS232-C, loop di corrente di 20 mA, oppure compatibilità TTL. La selezione di uno di essi si effettua per mezzo di ponticelli presenti sulla scheda del Nanocomputer. Per istruzioni su come effettuare tali collegamenti vi rimandiamo al *Manuale Tecnico del Nanocomputer.* La configurazione generale del connettore J5, adibito all'I/O seriale, è riassunta nella Tabella 5-3.

CARATTERISTICHE GENERALI DEL SOFTWARE DI SISTEMA DEL NANOCOMPUTER

La Figura 5-5 riproduce un diagramma di flusso generale del sistema operativo del Nanocomputer. Esistono quattro possibili modalità per accedere a quest'ultimo.

RESET/POWER-ON:

Questo procedimento è eseguito tutte le volte che il Nanocomputer è resettato (mediante pressione del tasto RESET) oppure quando viene acceso. La prima funzione che s'incontra lungo questo cammino è l'esecuzione di un semplice test della memoria di lettura/scrittura utilizzata dal sistema operativo per il suo

Tabella 5-3. Connettore J5.

Pin	Denominazione	Funzione
1	REAR	Uscita per il controllo dello avanzamento del lettore di banda perforata (non utilizzato nel Nanocomputer)
2	RTXTTY	Loop di 20 mA per TTY seriale (loop di trasmissione) con il pin 5
3	RTXTTY	Come il pin 2
4	RREAR	Forma un loop con il pin 1 (non utilizzato nel Nanocomputer)
5	TXTTY	Loop di 20 mA per TTY seriale (loop di trasmissione) con il pin 2
6	DTR	Uscita di terminale pronto (non utilizzato nel Nanocomputer)
7	DSR	Ingresso per dati pronti (non utilizzato nel Nanocomputer)
8	RTS	Uscita per richiesta di trasmissione (non utilizzato nel Nanocomputer)
9	SICK	Uscita di baud rate per il modem (non utilizzato nel Nanocomputer)
10	GND	
11	GND	
12	+5V	
13	GND	
14	RRXTTY	Loop di 20 mA per TTY seriale (loop di ricezione) con il pin 18
15	TXTTL	Trasmissione seriale a livello TTL
16	RXTTL	Ricezione seriale a livello TTL
17	CTS	Clear per poter iniziare la trasmissione (non utilizzato nel Nanocomputer)
18	RXTTY	Loop di 20 mA per TTY seriale (loop di ricezione) con il pin 14

stack di dati. In seguito all'esito positivo di questo esame della RAM, il sistema operativo, in mancanza di istruzioni specifiche, stabilisce per difetto il formato dei byte sul nastro ed il baud rate (frequenza di trasmissione/ricezione) (600 baud). Infine, passa il controllo alla sequenza di operazioni che inizia con il salvataggio dello stato della CPU.

RST 38H:

Il secondo metodo per restituire il controllo al sistema operativo del Nanocomputer consiste in un restant alla locazione 0038H. In tutti e due i manuali (Vol. 1 e 3) avete già avuto occasione di assistere all'utilizzo di questo metodo per tornare da un programma, definito dall'utente, al sistema operativo. Si noti come non venga eseguita nessuna operazione nella fase preliminare della sequenza di passi che inizia con il salvataggio dello stato della CPU.

NMI:

Un terzo modo per accedere al sistema operativo del Nanocomputer è realizzato per mezzo di una interruzione non mascherabile. Questa può essere inviata alla CPU Z80 sia premendo il tasto BREAK sia ponendo momentaneamente a massa

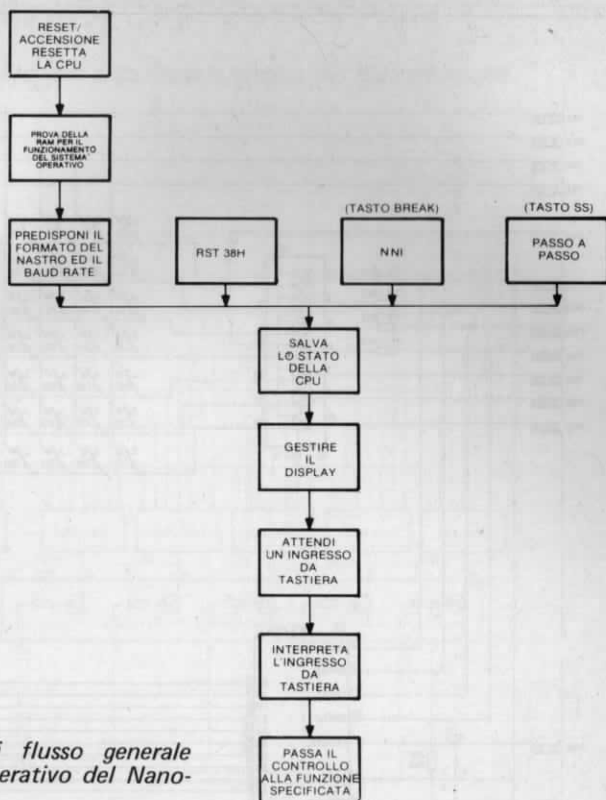


Figura 5-5. Diagramma di flusso generale del sistema operativo del Nanocomputer.

l'ingresso $\overline{\text{NMI}}$ della CPU. Nel capitolo 6 vi saranno proposti svariati esperimenti con impiego dell'ingresso NMI.

Dalla Figura 5-5 appare chiaro quale sia la differenza premendo il tasto BREAK oppure quello RESET. Essa consiste, nel caso del tasto RESET, nell'esecuzione di due funzioni in più, e precisamente il test di memoria e la determinazione del formato dei byte sul nastro e del baud rate. Eseguendo queste operazioni il sistema operativo inizializza i registri della CPU, azzerandoli. Proprio di questo abbiamo tenuto conto in numerose occcasioni, raccomandandovi di premere BREAK invece di RESET per trasferire senza distruggere il contenuto dei registri della CPU il controllo dal programma dell'utente al sistema operativo del Nanocomputer.

SINGLE STEP:

Il quarto ed ultimo modo per accedere al sistema operativo del Nanocomputer deriva dall'esecuzione passo per passo (single step) di un programma utente. Mentre procedete nell'esecuzione passo-passo (un'istruzione per volta) del programma che avete caricato in RAM, la CPU Z80 esegue, alternandole, prima la vostra istruzione e, subito dopo, diverse istruzioni del suo sistema operativo. Ad ogni trasferimento del controllo al sistema operativo, che segue l'esecuzione di ognuna delle vostre istruzioni, il controllo passa lungo quest'ultimo cammino.

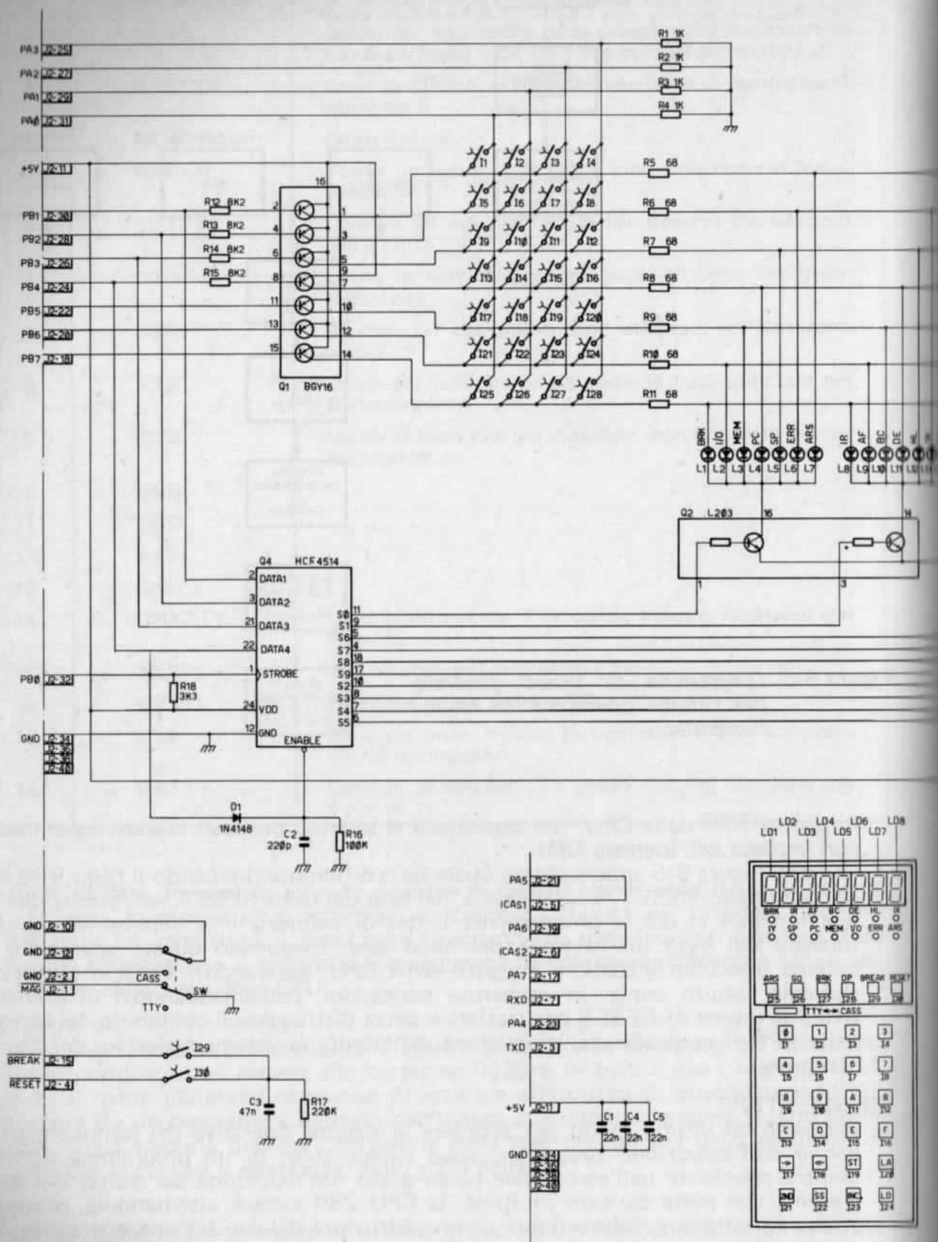
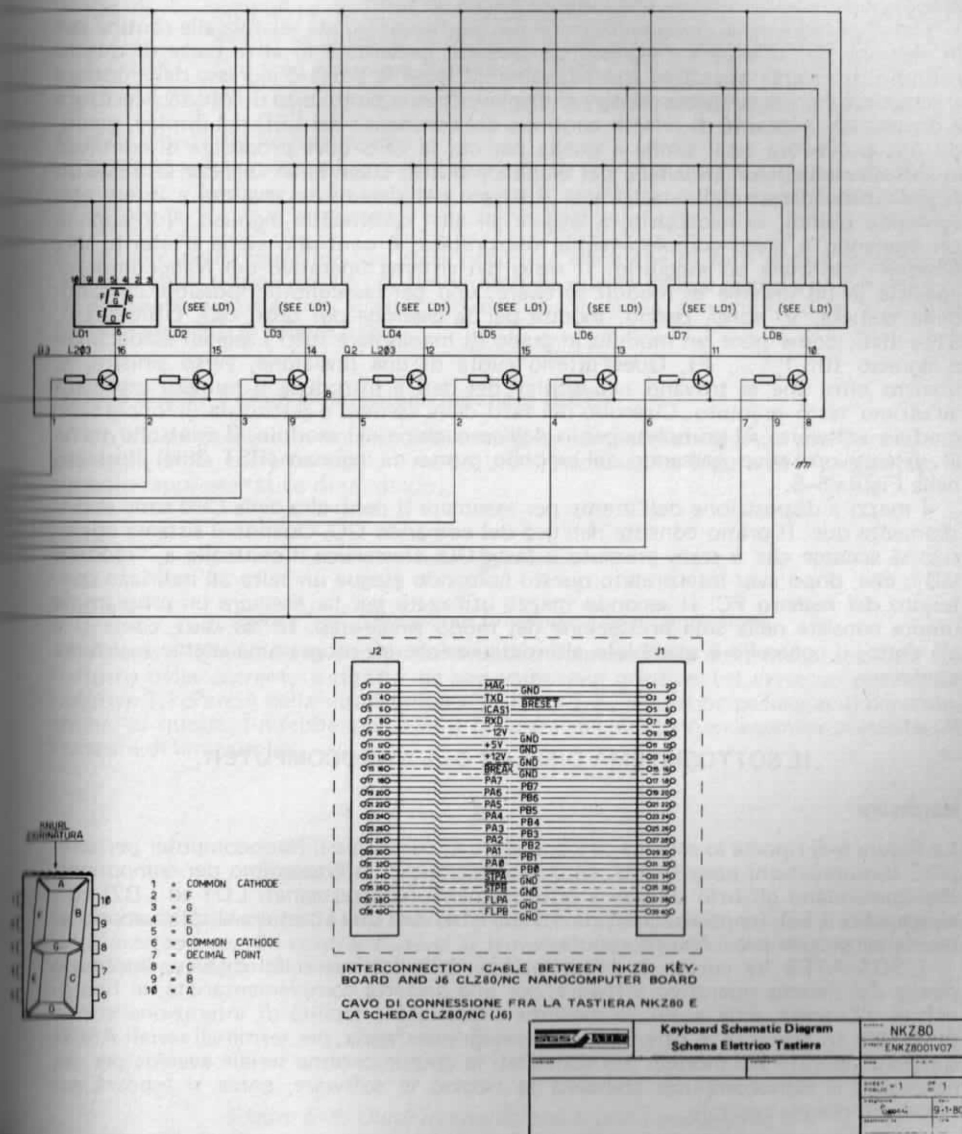

PA3 J2-25
PA2 J2-27
PA1 J2-29
PAØ J2-31
+5V J2-11
PB1 J2-30
PB2 J2-28
PB3 J2-26
PB4 J2-24
PB5 J2-22
PB6 J2-20
PB7 J2-18
R1 1K
R2 1K
R3 1K
R4 1K
R5 68
R6 68
R7 68
R8 68
R9 68
R1Ø 68
R11 68
R12 8K2
R13 8K2
R14 8K2
R15 8K2
Q1 BGY16
BRK I/O MEM PC SP ERR ARS
IR AF BC DE
L1 L2 L3 L4 L5 L6 L7
L8 L9 L1Ø L11
Q2 L2Ø3
Q4 HCF4514
DATA1
DATA2
DATA3
DATA4
STROBE
VDD
GND
ENABLE
PBØ J2-32
R18 3K3
GND
D1 1N4148
C2 22Øp
R16 1ØØK
GND J2-10
GND J2-12
GND J2-2
MAG J2-1
CASS
SW
TTY
BREAK J2-15
RESET J2-4
I29
I3Ø
C3 47n
R17 22ØK
PA5 J2-21
ICAS1 J2-5
PA6 J2-19
ICAS2 J2-6
PA7 J2-17
RXD J2-7
PA4 J2-23
TXD J2-3
+5V J2-11
C1 22n
C4 22n
C5 22n
GND
LD1 LD2 LD3 LD4 LD5 LD6 LD7 LD8
BRK IR AF BC DE HL IX
IY SP PC MEM I/O ERR ARS
ARS GO BRK DP BREAK RESET
TTY CASS

Figura 5-6. Schema dell'hardware della tastiera/display del Nanocomputer.



Una volta che la CPU Z80 è entrata nel sistema operativo, da uno qualsiasi dei precedenti quattro punti, esegue una serie di cinque funzioni base. Per prima cosa, salva lo stato di tutti i registri della CPU, accantonandone il contenuto nello stack di sistema. Nel seguito di questo capitolo saranno meglio precisate la dimensione, locazione e l'esatta utilizzazione delle aree RAM riservate ai dati ed allo stack del software di sistema. Successivamente il sistema operativo compone e fa apparire i dati appropriati sulla periferica tastiera/display. Come avrete certo osservato, la natura di ciò che appare sul display dipende dalla posizione dell'indicatore luminoso di selezione e dal contenuto del registro (dei registri) da visualizzare. La terza funzione consiste nell'attendere che l'utente prema un tasto.

Come si vedrà nella documentazione più particolareggiata relativa alle routine per la gestione del display e l'ingresso da tastiera, presentate in altra parte di questo capitolo, il sistema operativo alternativamente legge la porta d'ingresso dalla tastiera e scrive nella porta di uscita relativa al display. Questa alternanza di lettura e scrittura è dovuta alla necessità di refresh continuo del contenuto dei LED del display, secondo una procedura assai simile a quella per cui la CPU deve procedere di continuo al refresh della RAM dinamica del Nanocomputer. Quando in ingressi si rileva un segnale proveniente dalla tastiera, il software del sistema lo analizza e interpreta, tornando quindi, se necessario, a leggere gli altri caratteri in ingresso. Non appena un comando è stato completamente interpretato, il controllo viene trasferito alla funzione destinata ad eseguirlo. Il resto del sistema operativo del Nanocomputer consiste in un insieme di moduli software, uno per ciascuno dei possibili comandi della tastiera. Vi sono, perciò, moduli per la gestione dei tasti, GO, DUMP, LD, ST e INC, come pure un modulo in grado di manipolare tutti i segnali esadecimali d'ingresso (0, 1, . . ., F). Quest'ultimo ruota di una posizione, verso sinistra, le quattro cifre che si trovano nel display dei dati e introduce il numero associato all'ultimo tasto premuto. Ciascuno dei tasti della tastiera è gestito da un'opportuno modulo software. Al completamento dell'esecuzione del modulo, il controllo torna al sistema operativo passando dal secondo punto di ingresso (RST 38H) illustrato nella Figura 5-5.

I mezzi a disposizione dell'utente per assumere il controllo della CPU sono essenzialmente due. Il primo consiste nell'uso del comando GO. Quando il sistema operativo si accorge che è stato premuto il tasto GO, trasferisce il controllo a, "modulo GO", che, dopo aver interpretato questo comando esegue un salto all'indirizzo contenuto dal registro PC. Il secondo mezzo utilizzato per far eseguire un programma utente consiste nella sola prestazione del modo single-step. In tal caso, come si è già detto, il controllo è scambiato alternativamente tra programma utente e sistema operativo.

IL SOTTOSISTEMA DISPLAY DEL NANOCOMPUTER

Hardware

La Figura 5-6 riporta lo schema dell'hardware di cui si vale il Nanocomputer per semplici comunicazioni utente/CPU. In questa sezione ci interesseremo dei componenti che comandano gli otto display a sette segmenti (contrassegnati LD1 ed LD2) ed i quattordici LED (contrassegnati da L1 ad L14) dell'unità tastiera/display appositamente progettata per il Nanocomputer.

L'SGS-ATES ha curato la progettazione della tastiera e del display, insieme a quella del sistema operativo software, per una perfetta complementarietà, al fine di offrire all'utente, cioè a voi, la massima flessibilità e facilità di interazione con la CPU Z80. In aggiunta, è offerta in opzione un'interfaccia, per terminali seriali ASCII standard. Mentre nei modelli più sofisticati la comunicazione seriale avviene per via hardware, il Nanocomputer standard fa ricorso al software, come si esporrà nel seguito di questo capitolo.

I componenti che compaiono nello schema precedente operano in modo congiunto per espletare la duplice funzione di ingresso dai tasti verso la CPU e di uscita da quest'ultima verso i LED ed i display a sette segmenti. Prendiamo in considerazione per primi i componenti di uscita:

Q1: BGY16, una combinazione di sette transistori PNP isolati in un contenitore a 16 pin

Q2, Q3: L201, combinazione di sette transistori Darlington NPN isolati in un contenitore a 16 pin

Q4: HCF4514, latch a 4 bit/decodificatore da 4 a 16 linee, COS/MOS, avente la funzione di generare in uscita un 1 logico di selezione

L1-L14: Diodi ad emissione luminosa (LED)

LD1-LD2: Blocchi di display a sette segmenti (quattro display per blocco).

E, sulla scheda del Nanocomputer:

PIO Z80 N. 1: Le due parti di I/O parallelo del PIO Z80, costituiscono l'interfaccia di controllo, Porta B, per le linee contrassegnate in Figura 5-6, PB0-PB7.

Poichè un'analisi approfondita delle proprietà elettriche di tutti questi componenti richiederebbe troppo spazio, ci limiteremo in questa sede a ricordarne brevemente le proprietà più rilevanti, giusto per mettervi in grado di comprendere il circuito particolareggiato di Figura 5-6.

Diodi e display a diodi ad emissione luminosa (LED)

Un *diodo* consiste in un dispositivo elettronico a semiconduttori che permette il passaggio della corrente solo in modo unidirezionale. In Figura 5-7 è riportato il simbolo rappresentativo di un diodo.



Figura 5-7. Simbolo di un diodo a semiconduttore.

La freccia si riferisce alla convenzione di uso comune nella elettronica, secondo cui il flusso della corrente è diretto da un potenziale positivo (+) verso un potenziale negativo (-). Perciò nella situazione della Figura 5-8, si avrebbe passaggio di corrente, anche se questa finirebbe poi con il bruciare del diodo non essendo presente un resistore di limitazione.



Figura 5-8. Diodo con polarizzazione diretta.

Se un diodo è percorso da una corrente di valore apprezzabile, si dice che è *polarizzato direttamente.* Al contrario, se ci si trovasse nella situazione di Figura 5-9, non vi sarebbe passaggio di corrente.



Figura 5-9. Diodi in condizioni di non conduzione.

Trattando delle proprietà dei diodi, si incontrano spesso due importanti termini: *anodo* e *catodo.* La Figura 5-10 illustra, l'anodo e il catodo di un diodo comune.



Figura 5-10. Anodo e catodo di un diodo.

In base a queste definizioni di anodo e catodo, possiamo fare queste importanti affermazioni:

- Quando l'anodo è negativo rispetto al catodo, il diodo impedisce che passi corrente
- Quando l'anodo è positivo rispetto al catodo, il diodo permette il libero passaggio della corrente.

Un *diodo ad emissione luminosa* più semplicemente LED, non è altro che un diodo caratterizzato dalla proprietà di emettere luce se polarizzato direttamente e quindi, percorso da corrente. Il simbolo rappresentativo di un LED è riportato in Figura 5-11. Le estremità del LED sono le stesse di un diodo normale (si veda Figura 5-12).



Figura 5-11. LED.



Figura 5-12. Anodo e catodo di un LED.

Tutti i diodi, compresi i LED, sono dispositivi a semiconduttori piuttosto delicati, essendo soggetti a "bruciare", se la corrente che fluisce dall'anodo al catodo diventa eccessiva. E' dunque una regola tassativa, quando si opera con dei LED, quella di disporre un *resistore limitatore di corrente* tale da contenere il flusso di corrente che attraversa i LED entro limiti non pericolosi, oppure sufficienti a permettere la emissione luminosa da parte del LED. I resistori R5-R11 di Figura 5-6 sono tutti resistori utilizzati per la limitazione della corrente associati ai LED dei display a sette segmenti nonchè dei quattordici indicatori luminosi di selezione. In Figura 5-13 è riportato un esempio di un circuito a LED con resistore di 330 ohm per la limitazione della corrente.

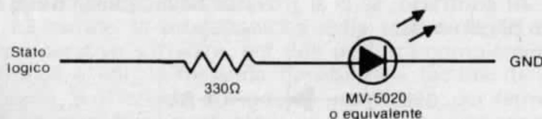


Figura 5-13. Circuito per indicatore luminoso.

Il LED più comune a luce rossa oppure verde consiste in un minuscolo wafer di semiconduttore, montato elettricamente su di una base metallica, alla quale è collegato un filo sottile (si veda la Figura 5-14).



Figura 5-14. Schema di principio di un LED.

L'insieme così ottenuto è poi incapsulato con plastica epoxy trasparente secondo una configurazione che fa sembrare in apparenza molto più grande la minuscola superficie di emissione. Si adotta, in altri termini, una "lente" in epoxy al fine di aumentare di circa due volte e mezzo il diametro della superficie luminosa.

I LED sono altresì realizzabili in forma di sbarretta o altre forme per essere impiegati in display di tipo diverso. Il display a sette segmenti di Figura 5-15 costituisce, appunto, un esempio di sette LED conformati a sbarretta.

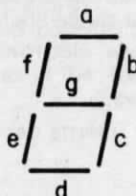


Figura 5-15. Display a sette segmenti.

Ognuno dei LED a sbarretta è associato ad una delle lettere a, b, c, d, e, f e g. E' facile verificare che, accedendo una determinata combinazione di tali segmenti, è possibile ottenere qualsiasi numero esadecimale. La corrispondenza tra i digit esadecimali e le combinazioni dei segmenti accesi è riportata qui sotto:

Digit esadecimale	Combinazione dei segmenti accesi
0	a,b,c,d,e,f,
1	b,c
2	a,b,d,e,g
3	a,b,c,d,g
4	b,c,f,g
5	a,c,d,f,g
6	a,c,d,e,f,g
7	a,b,c
8	a,b,c,d,e,f,g
9	a,b,c,f,g
A	a,b,c,e,f,g
b	c,d,e,f,g
C	a,d,e,f
d	b,c,d,e,g
E	a,d,e,f,g
F	a,e,f,g

Applicando queste nozioni, relative all'hardware, all'esame dello schema di Figura 5-6, si può notare come per l'accensione di uno qualsiasi dei quattordici LED sia sufficiente portare il suo anodo a +5V ed il suo catodo al potenziale di massa. Più precisamente, il collettore (si veda la nota sui transistori contenuta nel seguito) di un transistore PNP di BGY16 deve trovarsi nello stato logico 1 mentre il collettore di un transistore NPN Darlington di Q2 deve trovarsi nello stato logico 0. Per accendere uno qualsiasi dei 14 led dell'unità tastiera/display. Similmente, per rendere luminosa la desiderata combinazione di led dei display a sette segmenti di LD1 ed LD2, occorre portare nello stato logico 1 i collettori degli appropriati transistori PNP di BGY16 e nello stato logico 0 il collettore dell'appropriato NPN Darlington di Q2 o di Q3. Si faccia attenzione al fatto che i transistori di BGY16 risultano collegati alla porta B del PIO N. 1. Questo fa intuire che sicuramente il bus dei dati contribuisce a determinare quali sono i segmenti e/o i led che si dovranno accendere.

Transistori

I transistori possono essere distinti in due tipi fondamentali, transistori PNP ed NPN. Queste sigle si riferiscono alla particolare disposizione interna con cui il materiale semiconduttore viene disposto per ottenere un transistore. I materiali interessati sono:

Semiconduttore di Tipo N — possiede un eccesso di elettroni ed è caratterizzato da una carica libera negativa;

Semiconduttore di Tipo P — possiede elettroni in difetto (ovvero "lacune" in eccesso) ed è caratterizzato da una carica libera positiva.

La Figura 5-16 illustra la disposizione interna dei materiali di tipo P e di tipo N per transistori PNP ed NPN.



Figura 5-16. Giunzioni di tipo N e P.

Negli schemi elettrici i due tipi di transistori sono rappresentati mediante i simboli di Figura 5-17.

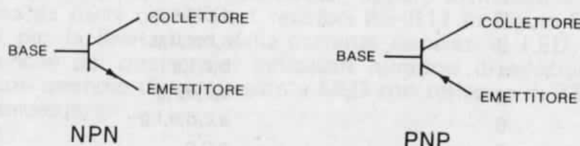


Figura 5-17. Simboli dei transistori.

Si osservi che i terminali collegati alle differenti sezioni del materiale semiconduttore sono contrassegnate come *emettitore, base e collettore*. La direzione della freccia associata all'emettitore indica il tipo di transistore rappresentato.

I transistori possono essere considerati come interruttori controllati elettricamente, che possono essere "aperti" o "chiusi" variando la corrente elettrica che passa nella

base. Affinchè un transistore sia "aperto", ossia permetta il passaggio di corrente, esso deve essere polarizzato correttamente, presentare cioè, un'opportuna relazione fra i potenziali di base, emettitore e collettore. Questa relazione è differente a seconda che si tratti di un transistore PNP oppure di uno NPN. La Figura 5-18 riporta la corretta relazione dei potenziali per transistori di entrambi i tipi in fase di conduzione.

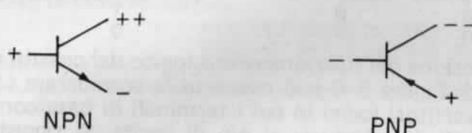


Figura 5-18. Transistori PNP e NPN in conduzione.

Si osservi che la freccia associata all'emettitore sta ad indicare un comportamento analogo a quello di un diodo, nel senso che, agli effetti del passaggio di corrente, il suo anodo deve essere positivo rispetto al catodo. Per bloccare un transistore è sufficiente annullare la sua corrente di base. La Figura 5-19 illustra transistori NPN e PNP che non conducono (in interdizione).

Proviamo ora ad applicare questi concetti alla funzione dei transistori Q1, Q2 e Q3 dello schema di Figura 5-6.

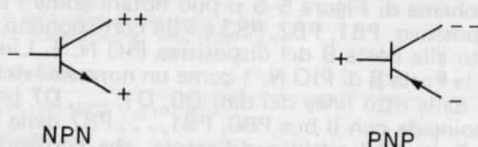


Figura 5-19. Transistori NPN e PNP interdetti.

TRANSITORI PNP Q1:

L'emettitore di ognuno dei sette transistori PNP di BGY16 è collegato a +5 volt. Ciascuno di essi è, perciò, polarizzato correttamente se la linea PBn (dove n = 0, . . ., 7), ad esso associata, si trova nello stato logico 0 oppure è bloccato (interdetto) se la linea PBn si trova nello stato logico 1. Se un transistore e collettore è trascurabile (0,1 o 0,2 volt) ed in questo caso il collettore si trova in pratica allo stesso potenziale dell'emettitore. Se, al contrario, il transistore è interdetto, cioè non conduce corrente, esiste una differenza di potenziale tra emettitore (nello stato logico 1) e collettore. Ne consegue che, per i transistori PNP Q1 del circuito di I/O del Nanocomputer (Figura 5-6), si può assumere come specchio del loro comportamento il seguente prospetto semplificato:

Emettitore	Base	Collettore
1	0	1
1	1	0

TRANSISTORE NPN Q2 e Q3:

L'emettitore di ognuno dei dieci transistori NPN è collegato a massa. Il transistore è, perciò, polarizzato correttamente se la base si trova nello stato logico 1, e bloccato (interdetto) se essa si trova nello stato logico 0. Analogamente a quanto suc-

cede per i transistori PNP, collettore ed emettitore si trovano praticamente allo stesso potenziale quando il transistore conduce, mentre esiste una differenza di potenziale tra i due quando il transistore è interdetto. Una tabella della verità di un transistore NPN, con qualche semplificazione, è costituita dal seguente prospetto:

Emettitore	Base	Collettore
0	0	1
0	1	0

Ai fini della comprensione del funzionamento logico dei circuiti di I/O del Nanocomputer dello schema di Figura 5-6 può essere utile considerare i transistori di Q1, Q2 e Q3 come degli invertitori logici in cui i terminali di base corrispondono ai pin di ingresso ed i terminali di collettore ai pin di uscita. Si ricordi, comunque, che il comportamento generale dei transistori merita una spiegazione molto più complessa di quella a cui ci limitiamo in questa sede. Per uno studio più compleso dei transistori vi rimandiamo ad altre fonti.

Latch a 4 bit - Decodificatore da 4 a 16 linee (selezione attiva alta)
COS/MOS HCF4514B

Il chip HCF4514B consiste in un latch a 4 bit abbinato ad un decodificatore da 4 a 16 linee. La configurazione dei pin e la tabella della verità relativa alla sezione del decodificatore sono riportate nella Tavola 5-1.

Riferendosi allo schema di Figura 5-6 si può notare come i quattro ingressi, alla sezione latch del dispositivo, PB1, PB2, PB3 e PB4 corrispondoo a quattro delle otto linee del bus associato alla Porta B del dispositivo PIO N. 1. Limitiamoci, per il momento a considerare la Porta B di PIO N. 1 come un normale latch a otto bit, i cui ingressi sono costituiti dalle otto linee dei dati D0, D1, . . ., D7 provenienti dalla CPU Z80 e la cui uscita coincide con il bus PB0, PB1, . . , PB7 della Porta B di PIO N. 1. PB0 è collegato al pin 1 del latch/decodificatore, che corrisponde, per l'appunto, all'ingresso di strobe del latch a 4 bit. Una transizione da alto a basso di PB0 (da considerarsi equivalente a D0) determina il latch di PB1-PB4 (cioè D1-D4). Una volta sottoposto a latch, i valori presenti sulle linee PB1-PB4 sono inviati agli ingressi del decodificatore da 4 a 16 linee dell'HCF4514B, preposto alla selezione delle 16 linee di uscita, una delle quali sarà portata dal suo normale stato logico 0, allo stato logico 1. Poichè ognuna delle prime 10 uscite del decodificatore, S0-S9, è collegata alla base di un transistore Darlington, capace a sua volta di polarizzare direttamente un LED o un display a sette segmenti, PB1-PB4 provvede a selezionare il display che si deve illuminare. Poichè il latch del chip HCF4514B conserva i dati PB1-PB4 per la selezione del display, la linea di uscita del decodificatore, precedentemente selezionata, si mantiene alta sintantochè non sia riportata esplicitamente allo stato basso mediante l'immissione nel latch di nuovi dati di selezione oppure quando si inibiscono le uscite del decodificatore (si veda il prossimo paragrafo).

Il pin 23 del latch/decodificatore corrisponde ad un ingresso inibit (di inibizione) che, quando si trova nello stato logico 1, disattiva tutte e 16 le uscite del decodificatore, portandole allo stato logico 0.

Si osservi come alla linea PB0 siano collegati sia il pin 1 che quello 23. La presenza del resistore R16 e del condensatore C2 fa sì che l'effetto delle variazioni dello stato logico sulla linea PB0 (e di conseguenza al pin 1) si manifesti al pin 23 con un certo ritardo. Il diagramma della temporizzazione, riportato in Figura 5-20, illustra l'influenza di R16 e C2 sulla risposta del pin 23 (ingresso inhibit) agli impulsi presenti sulla linea P0 messa a confronto con quella del pin 1 (ingresso di strobe).

Affinchè il pin 23 raggiunga uno stato logico 1 avvertibile è necessario che PB0 si mantenga alto per un intervallo di tempo relativamente lungo. Ne consegue che un impulso positivo (transizione 0 - 1 - 0) convenientemente rapido, verificantesi su

Tavola 5-1. Caratteristiche del Latch/Decodificatore HCF4514B.

PRELIMINARY DATA

4-BIT LATCH/4-TO-16 LINE DECODER:

HCC/HCF 4514B OUTPUT "HIGH" ON SELECT
HCC/HCF 4515B OUTPUT "LOW" ON SELECT

- QUIESCENT CURRENT SPECIFIED TO 20V
- MAX. INPUT LEAKAGE CURRENT 1 μA @ 18V (FULL PACKAGE - TEMP. RANGE)
- STROBED INPUT LATCH
- INHIBIT CONTROL

The **HCC 4514B/HCC 4515B** (extended temperature range) and the **HCF 4514B/HCF 4515B** (intermediate temperature range) are monolithic integrated circuits available in 24-lead dual in-line plastic and ceramic slam package. The **HCC/HCF 4514B/4515B** consisting of a 4-bit strobed latch and a 4 to 16 line decoder. The latches hold the last input data presented prior to the strobe transition from 1 to 0. Inhibit control allows all outputs to be placed at 0 (**HCC/HCF 4514B**) or 1 (**HCC/HCF 4515B**) regardless of the state of the data or strobe inputs. The decode truth table indicates all combinations of data inputs and appropriate selected outputs.

ABSOLUTE MAXIMUM RATINGS

V_{DD}*	Supply voltage	-0.5 to 20	V
V_I	Input voltage	-0.5 to V_{DD} +0.5	V
I_I	DC input current (any one input)	± 10	mA
P_{tot}	Total power dissipation (per package)	200	mW
	Dissipation per output transistor for T_{op} = full package-temperature range	100	mW
T_{op}	Operating temperature: for **HCC** types	-55 to 125	°C
	for **HCF** types	-40 to 85	°C
T_{stg}	Storage temperature	-65 to 150	°C

* All voltage values are referred to V_{SS} pin voltage

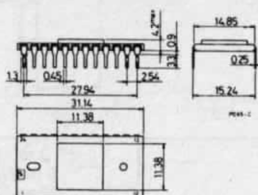
ORDERING NUMBERS:

HCC 45XX BD for dual in-line ceramic slam package
HCF 45XX BD for dual in-line ceramic slam package
HCF 45XX BE for dual in-line plastic package

MECHANICAL DATA

dimensions in mm

Dual in-line ceramic slam package for HCC/HCF 45XX BD



Dual in-line plastic package for HCF 45XX BE

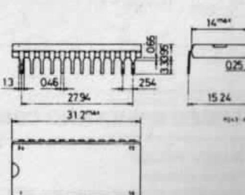
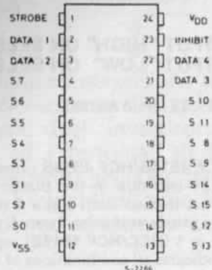
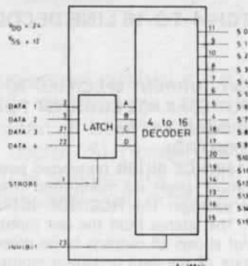


Tavola 5-1. Caratteristiche del Latch/Decodificatore HCF4514B (seguito).

NNECTION DIAGRAM



FUNCTIONAL DIAGRAM



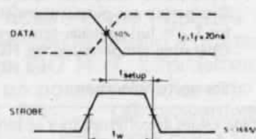
DECODER TRUTH TABLE*

INHIBIT	DATA INPUTS				SELECTED OUTPUT HCC/HCF 4514B= Logic 1 (High) HCC/HCF 4515B= Logic 0 (Low)
	D	C	B	A	
0	0	0	0	0	S0
0	0	0	0	1	S1
0	0	0	1	0	S2
0	0	0	1	1	S3
0	0	1	0	0	S4
0	0	1	0	1	S5
0	0	1	1	0	S6
0	0	1	1	1	S7
0	1	0	0	0	S8
0	1	0	0	1	S9
0	1	0	1	0	S10
0	1	0	1	1	S11
0	1	1	0	0	S12
0	1	1	0	1	S13
0	1	1	1	0	S14
0	1	1	1	1	S15
1	X	X	X	X	All Outputs= 0, HCC/HCF 4514B All Outputs= 1; HCC/HCF 4515B

X = Don't Care *Strobe = 1
1 = high
0 = low

WAVEFORMS

Setup time and strobe pulse width

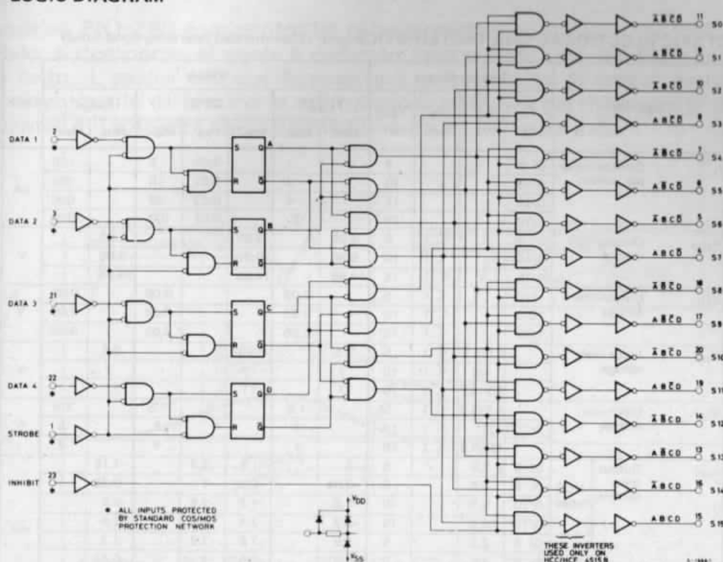


RECOMMENDED OPERATING CONDITIONS

V_{DD}	Supply voltage	3 to 18	V
V_I	Input voltage	0 to V_{DD}	V
T_{op}	Operating temperature: for **HCC** types	-55 to 125	°C
	for **HCF** types	-40 to 85	°C

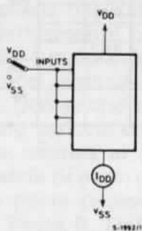
Tavola 5-1. Caratteristiche del Latch/Decodificatore HCF4514B (seguito).

LOGIC DIAGRAM

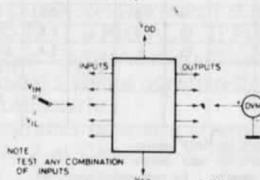


TEST CIRCUITS

Quiescent device current



Noise immunity



Input leakage current

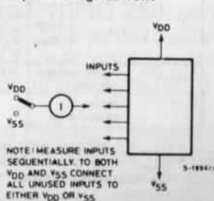


Tavola 5-1. Caratteristiche del Latch/Decodificatore HCF4514B (seguito).

STATIC ELECTRICAL CHARACTERISTICS (over recommended operating conditions)

Parameter			Test conditions				Values							Unit
			V_I (V)	V_O (V)	$\lvert I_O \rvert$ (μA)	V_{DD} (V)	T_{Low}*		25°C			T_{High}*		
							Min.	Max.	Min.	Typ.	Max.	Min.	Max.	
I_L	Quiescent supply current		0/ 5			5		1		0.02	5		150	μA
			0/10			10		2		0.02	10		300	
			0/15			15		4		0.02	20		600	
			0/20			20		20		0.04	100		3000	
V_{OH}	Output high voltage		0/ 5		< 1	5	4.95		4.95			4.95		V
			0/10		< 1	10	9.95		9.95			9.95		
			0/15		< 1	15	14.95		14.95			14.95		
V_{OL}	Output low voltage		5/0		< 1	5		0.05			0.05		0.05	V
			10/0		< 1	10		0.05			0.05		0.05	
			15/0		< 1	15		0.05			0.05		0.05	
V_{IH}	Input high voltage			0.5/4.5	< 1	5	3.5		3.5			3.5		V
				1/9	< 1	10	7		7			7		
				1.5/13.5	< 1	15	11		11			11		
V_{IL}	Input low voltage			4.5/0.5	< 1	5		1.5			1.5		1.5	V
				9/1	< 1	10		3			3		3	
				13.5/1.5	< 1	15		4			4		4	
I_{OH}	Output drive current	HCC types	0/ 5	2.5		5	-2		-1.6	-3.2		-1.15		mA
			0/ 5	4.6		5	-0.64		-0.51	-1		-0.36		
			0/10	9.5		10	-1.6		-1.3	-2.6		-0.9		
			0/15	13.5		15	-4.2		-3.4	-6.8		-2.4		
		HCF types	0/ 5	2.5		5	-1.8		-1.6	-3.2		-1.3		
			0/ 5	4.6		5	-0.61		-0.51	-1		-0.42		
			0/10	9.5		10	-1.5		-1.3	-2.6		-1.1		
			0/15	13.5		15	-4		-3.4	-6.8		-2.8		
I_{OL}	Output sink current	HCC types	0/ 5	0.4		5	0.64		0.51	1		0.36		mA
			0/10	0.5		10	1.6		1.3	2.6		0.9		
			0/15	1.5		15	4.2		3.4	6.8		2.4		
		HCF types	0/ 5	0.4		5	0.61		0.51	1		0.42		
			0/10	0.5		10	1.5		1.3	2.6		1.1		
			0/15	1.5		15	4		3.4	6.8		2.8		
I_{IH}, I_{IL}**	Input leakage current		0/18			18		±0.1		$\pm 10^{-5}$	±0.1		± 1	μA
C_I**	Input capacitance									5	7.5			pF

* T_{Low} = - 55°C for **HCC** device; - 40°C for **HCF** device.
* T_{High} = +125°C for **HCC** device; + 85°C for **HCF** device.
The Noise Margin for both "1" and "0" level is: 1V min. with V_{DD} = 5V; 2V min. with V_{DD} = 10V; 2.5V min. with V_{DD} = 15V
** Any input

PB0, non sarà avvertito al pin 23 e costituirà unicamente il segnale di strobe per un nuovo codice a quattro bit per il display rilevato al pin 1 in concomitanza della transizione alto-basso. L'importanza pratica di questo comportamento consiste nel fatto che i nuovi digit del display possono essere selezionati senza disabilitare l'uscita del decodificatore, ottenendo in tal modo un display esente da intermittenze, in cui si noterà di meno il continuo refresh del display.

Il dispositivo PIO Z80

Il dispositivo PIO Z80 è un controller programmabile complesso per interfaccia di I/O parallelo, a due porte, al quale è dedicato interamente uno dei prossimi capitoli di questo testo. L'analisi che qui faremo sarà perciò molto sintetica, avendo come unico obiettivo quello di fornirvi le informazioni essenziali perchè possiate comprendere l'hardware e il software che riguardano i circuiti di Figura 5-6.



Figura 5-20. Effetto di R16 e C2 sulla Risposta del pin di inibizione alle Transizioni sulla linea P0 (ingresso STB).

Sulla scheda del Nanocomputer sono presenti due dispositivi PIO Z80, contrassegnati rispettivamente con PIO N. 1 e PIO N. 2. Il PIO N. 1 è stato adibito all'espletamento delle funzioni di I/O con l'unità tastiera/display del Nanocomputer, mentre il PIO N. 2 non viene utilizzato dal sistema oprativo del Nanocomputer, perciò è disponibile per l'utente che voglia utilizzarlo.

In questa sezione concentreremo la nostra attenzione sul PIO N. 1, rimandando la trattazione relativa al PIO N. 2 al Capitolo 7, dove avremo occasione di effettuare tutta una serie di esperimenti.

Le due porte presenti in un dispositivo PIO sono di norma contrassegnate come Porta A e Porta B. Un'adeguata programmazione del PIO permette di selezionare per ciascuna delle porte uno qualsiasi dei quattro modi di funzionamento. Nel caso si voglia selezionare il modo di funzionamento della Porta A e della Porta B la programmazione implica l'invio di una ben definita stringa di byte ad una porta di controllo associata alla Porta A e alla Porta B. Le porte di I/O associate al PIO N. 1 sono quattro:

Porta	Codice dispositivo
Porta A, dati	04
Porta B, dati	05
Porta A, controllo	06
Porta B, controllo	07

I codici dispositivo 8, 9, A e B sono associati in modo analogo al PIO N. 2. I quattro modi di funzionamento delle Porte A e B sono:

Modo 0: Uscita Byte:	Per uscita in parallelo pilotata da interruzione.
Modo 1: Ingresso Byte:	Per ingresso in parallelo pilotata da interruzione.
Modo 2: Trasferimento bidirezionale:	Per I/O bidirezionale, disponibile solamente sulla Porta A.
Modo 3: Control Mode (Modo controllato):	Sia per ingresso che per uscita. Ciascuno degli otto bit della porta è da interpretarsi come bit in ingresso oppure in uscita. Il dispositivo PIO si comporta fondamentalmente da latch per i bit in uscita e da buffer/latch per quelli in ingresso. (Nel Modo Controllato sono insiste delle possibilità addizionali molto interessanti, delle quali avremo occasione di occuparci nel seguito).

Mediante una routine del sistema operativo del Nanocomputer, avviene l'inizializzazione, ossia la programmazione del PIO N. 1 in modo da far funzionare nel modo Controllato (Modo 3) ambedue le Porte A e B, con tutti i bit della Porta B definiti come bit in uscita e quelli della Porta A definiti nel modo seguente: 10001111. Perciò i bit D0, D1, D2, D3 e D7 corrispondono a bit di ingresso, mentre i bit D4, D5 e D6 saranno altrettanti bit di uscita. Il risultato che si ottiene è di aver predisposto il PIO N. 1 in modo tale che un'istruzione OUT (05H),A porrebbe tutti i 7 bit del bus dei dati dello Z80 sul bus dei dati PB0-PB7 della Porta B, mentre con un'istruzione OUT (04H),A D4, D5 e D6 finirebbero su PA4, PA5 e PA6 del bus dei dati della Porta A. Un'istruzione di ingresso IN per la Porta B sarebbe priva di significato in quanto tutte le linee del suo bus dei dati sono state selezionate per l'uscita. Per altro con IN A, (04H) si avrebbe la lettura in accumulatore dei dati presenti sulle linee PA0, PA1, PA2, PA3 e PA7.

Software

Guardate allo schema di Figura 5-6 e osservate i punti di collegamento con il bus dei dati della Porta B, ossia PB0-PB7. Le linee della Porta A saranno considerate più avanti quando esamineremo l'ingresso dalla tastiera del Nanococmputer. Immaginiamo questa seguenza in eventi:

1. I bit PB1-PB4 sono posti a 0000 mediante le istruzioni:

```
XOR A
OUT (05H),A
```

2. Il bit PB0 subisce una doppia transizione in base alla sequenza di istruzioni:

```
INC A
OUT (05H),A
DEC A
OUT (05H),A
```

Si noti che i bit D1-D4 (in realtà D1-D7) rimangono inalterati mentre D0 (inizialmente allo 0 logico dopo essere stato precedentemente azzerato) subisce una transizione basso-alto-basso, che determina la stessa transizione per PB0, con la conseguenza finale che (in concomitanza della transizione alto-basso) PB1-PB4 sono immessi nel latch di HCF4514B. Risulta così attivata la linea S0

(pin 11) di uscita del decodificatore e, poichè è collegata alla base del transistore NPN Darlington associata ai sette LED, ne deriva che quel blocco di led è stato selezionato come display.

3. Si setta l'accumulatore in modo da provocare l'accensione del led MEM. Per far questo occorre tenere presente il comportamento dei transistori interessati. In conseguenza dell'1 logico della sua base, il transistore NPN Darlington di Q2 ha il collettore nello stato logico 0.
   Per polarizzare direttamente il led dell'indicatore luminoso di selezione MEM, il collettore del transistore PNP di Q1 ad esso associato deve trovarsi nello stato logico 1 (in modo che l'anodo risulti positivo rispetto al catodo). Tutti gli altri led non devono essere polarizzati, ossia i collettori di tutti gli altri transistori di Q1 devono trovarsi nello stato logico 0. Sappiamo che, affinchè i collettori si trovino in tale stato, i valori associati alle basi corrispondenti di ciascuno dei transistori devono essere quelli del *livello logico opposto.* In altri termini, affinchè MEM sia acceso, i bit in uscita su PB1-PB7 devono essere:

PB7:	1 logico perchè	L1,	indicatore BRK,	sia spento
PB6:	1 logico perchè	L2,	indicatore I/O,	sia spento
PB5:	0 logico perchè	L3,	indicatore MEM,	sia ACCESO
PB4:	1 logico perchè	L4,	indicatore PC,	sia spento
PB3:	1 logico perchè	L5,	indicatore SP,	sia spento
PB2:	1 logico perchè	L6,	indicatore ERR,	sia spento
PB1:	1 logico perchè	L7,	indicatore ARS,	sia spento

   Rimane da osservare che PB0 non può passare nello stato alto se non per intervalli di tempo molto brevi, dato che esso è collegato in serie con l'ingresso di inhibit dell'HCF4514, *attivo* appunto nello stato *alto*. Si useranno quindi le seguenti istruzioni per porre i valori desiderati sul bus di uscita della Porta B:

```
LD   A,DEH
OUT  (05H),A
```

   Il latch del dispositivo latch/decodificatore HCF4514B consente che il display desiderato (in questo caso il primo blocco di sette led) continui a rimanere selezionato, mentre il bus dei dati della Porta B (precedentemente usato per selezionare i LED del display) viene riutilizzato per eseguire un compito del tutto differente dalla selezione del display di uscita; precisamente la determinazione dei contenuti del display.

Quella appena descritta è l'esatta sequenza di operazioni eseguite dalla routine di gestione dell'uscita del Nanocomputer per la visualizzazione dell'indicatore luminoso e di cifre esadecimali. E se, per esempio, voleste far accendere due led, poniamo I/O e MEM? La risposta è semplice: la procedura da seguire è di per sé uguale alla precedente con la differenza che il byte finale dell'istruzione di uscita sarebbe 9E al posto di DE, per la corretta polarizzazione di entrambi i led I/O e MEM. E se voleste far accendere sia il led MEM che uno o più display a sette segmenti o un led dell'altro gruppo di sette? Il problema sarebbe qui di tutt'altro genere. Limitarsi infatti a cambiare il byte finale dell'istruzione di uscita non servirebbe al nostro scopo per due motivi: (1) i display che devono essere selezionati "contemporaneamente", o dare almeno questa *impressione* sono due e (2) due sono anche i byte di dati che devono essere posti in uscita, uno per l'accensione del LED MEM e il secondo per quella del secondo display. Com'è possibile ottenere questo risultato?

In primo luogo il circuito di Figura 5-6 *non* è in grado di attivare due display nello *stesso* preciso istante essendo il decodificatore in grado di attivare ogni volta solo le uscite UNO oppure ZERO, *senza* che vi siano *altre* possibilità! Il circuito, tuttavia, *è in grado* di attivare due display in un intervallo di tempo impercettibile. Senza, cioè, che l'occhio umano possa avvedersene, la CPU ed il circuito associato possono combinare la loro azione in modo da attivare ripetutamente a cicli alterni un led e/o un digit del display e poi l'altro. In tal modo sembra che entrambi i display siano *attivati contemporaneamente*. Anche nel caso di una visualizzazione estesa a tutti e dieci i possibili blocchi di led e display a sette segmenti la CPU può muoversi ciclicamente su ciascuno di essi, attivando alternativamente i diversi display e l'invio in uscita del byte di dati necessario a far accendere i led e/o segmenti appropriati. Il ciclo che permette di ricoprire tutti e dieci i display è talmente rapido che ogni singolo display è *rigenerato* con una frequenza talmente elevata che un occhio umano non riesce a percepire nemmeno un tremolio nel display! Una interessante caratteristica, sfruttata in pratica da tutti i progettisti di dispositivi di visualizzazione quali televisori, CRT e monitor, è la persistenza dell'immagine sulla retina dell'occhio umano. E' sufficiente che un carattere o un'immagine, sia rigenerato anche solo 60 volte al secondo affinchè l'occhio umano veda l'immagine fissa sul display. Il numero di "riscritture" del carattere o dell'immagine è detto *refresh rate* (frequenza di rigenerazione).

Il seguente programma illustra un semplice sistema di comando del display, in grado di accendere più di un'unità del display per volta. L'accumulatore funge da selettore del display, mentre il registro B contiene il byte di dati da visualizzare. Si noti come l'accumulatore sia incrementato non una ma due volte al termine di ogni loop (appena prima della chiamata della subroutine di DELAY, ritardo). I bit che determinano effettivamente la selezione del display sono D1-D7 (cioè PB1-PB7). Dal momento che un doppio incremento di D0-D7 equivale al semplice incremento di D1-D7 occorre utilizzare due, e non una sola, istruzioni di incremento. Osservate che il contenuto del registro B rimane invariato per tutto il programma. Perciò su tutti i dieci display viene visualizzato lo stesso byte di dati. I primi due blocchi di sette led presenteranno, naturalmente, un aspetto diverso da quello dei display a sette segmenti, ma il led o segmento che si accenderà in quest'ultimo caso sarà uno per ciascuno di essi. Dallo schema si può vedere che si accenderanno L1, L8 ed il segmento A di ciascuno degli otto display a sette segmenti.

Per poter accendere, uno alla volta, tutti i dieci LED e i display a sette segmenti il registro D conta all'indietro da 10 a 0. Al compimento del processo di selezione D, avrà nel frattempo, raggiunto il valore zero. Questo determinerà un salto all'indietro proprio all'inizio della routine di gestione del display (istruzione BEGIN), dove si

```
BEGIN:    LD D, 0AH        ; Inizializza il contatore
          XOR A            ; Inizializza il selettore del display
          LD BC,7E05H      ; Inizializza il byte dei dati a 7E e della porta di uscita a 05

OUTPUT:   OUT (C),A        ; Poni in uscita il byte di selezione del display
          INC A            ; verso il decodificatore/latch mediante
          OUT (C),A        ; variazioni imposte al bit D0
          DEC A
          OUT (C),A
          OUT (C),B        ; Poni in uscita il byte dei dati
          INC A            ; Aggiorna il registro A per puntare
          INC A            ; al display successivo
          CALL DELAY       ; Ritardo
          DEC D            ; Decrementa D per controllare se il ciclo è concluso
          JR NZ, OUTPUT    ; Se D non è 0, continua nel ciclo interno
          JR BEGIN         ; Se D è uguale a 0, ricomincia dal principio.
```

procederà alla reinizializzazione di tutti i registri per la ripetizione del medesimo ciclo attraverso i display. Al termine di questo capitolo voi stessi proverete in un esperimento alcune modifiche a questo programma.

Più avanti, in una sezione dedicata al software del sistema operativo del Nanocomputer, avremo occasione di esaminare il modo di gestione del display, utilizzato dal sistema operativo, per porre in uscita le diverse visualizzazioni che vedete quando premete i tasti della tastiera. Prima di affrontare questo argomento, proviamo a studiare i circuiti di ingresso riportati dallo schema di Figura 5-6 ed il software necessario per la loro gestione.

INGRESSO DAL SOTTOSISTEMA TASTIERA DEL NANOCOMPUTER

Hardware

Per vostra comodità, riproduciamo in Figura 5-21 lo schema dell'hardware della tastiera/display del Nanocomputer.
I componenti di questo circuito interessati alle operazioni di ingresso della tastiera del Nanocomputer sono:

Q1: BGY16, insieme di sette transistori PNP isolati e montati in un contenitore a 16 pin;

I1 I28: 28 tasti della tastiera del Nanocomputer.

Come si può osservare, i componenti relativi all'ingresso sono in numero inferiore rispetto a quelli per l'uscita. Poichè dei sette transistori di BGY16 si è già avuto modo di parlare a proposito del sottosistema display del Nanocomputer, non resta che esaminare il gruppo di 28 tasti che compongono la tastiera del Nanocomputer.

Tastiera a 28 tasti: I1-I28

Dal punto di vista logico, la tastiera del Nanocomputer è disposta secondo un insieme di sette righe per quattro colonne di tasti. Si noti che le file presentano una corrispondenza biunivoca con le linee di uscita PB1-PB7, mentre le colonne si trovano in analoga corrispondenza con le linee di ingresso PA0-PA3. Il rilevamento, la struttura antirimbalzo e l'identificazione dei tasti premuti sono affidati a diverse routine software, facenti parte del sistema operativo del Nanocomputer. Per antirimbalzo intendiamo l'introduzione di un intervallo di ritardo nella routine software che, (a), rileva l'abbassamento di un tasto, e (b), attende sino a che il tasto si stabilizzi o nella posizione di tasto abbassato, oppure in quella di tasto rilasciato. Poichè i tasti non sono dotati di circuiti hardware antirimbalzo, la transizione dello stato di tasto premuto a quello opposto, e viceversa, non avviene istantaneamente. Esiste perciò un breve intervallo di tempo durante il quale i contatti metallici "rimbalzano" dallo stato aperto a quello chiuso prima di arrestarsi in uno stato definitivo di aperto oppure chiuso. Un espediente molto agevole ed economico per eliminare tutti i problemi che si possono presentare con dei tasti privi di hardware antirimbalzo è costituito da un breve ciclo di ritardo, introdotto nel software adibito all'identificazione tasto premuto. In tal modo la lettura dei tasti da parte del software avviene sempre quando questi hanno raggiunto una posizione di stabilità che è naturalmente indispensabile per garantire l'affidabilità dei dati introdotti da tastiera.

Software

Il sistema operativo del Nanocomputer rileva in due passi successivi i tasti premuti. Durante la prima fase, per scoprire se *un qualsiasi* tasto è stato premuto, viene eseguita una routine di nome CHECKB. Se in base a questa prima indagine, risulta che è stato premuto un tasto, si passa alla seconda fase, durante la quale, con una routine denominata KBSCAN, si determina:

a) Se è stato premuto esattamente UN tasto. Nel caso che sia stato premuto contemporaneamente più di un tasto si ignorano tutti i tasti premuti.

b) Nel caso che sia stato premuto un solo tasto, di quale tasto si tratta? I tasti sono numerati da 1 a 28, per cui la risposta che si ottiene consiste in un numero compreso tra 1 e 28.

Entrambe le routine CHECKB e KBSCAN utilizzano la stessa tecnica di base per il rilevamento dei tasti premuti. Uno o più bit di PB1-PB7 sono posti nello stato logico 0, provocando uno stato logico 1 per gli associati collettori dei transistori di Q1. Dal modo con cui i tasti sono rappresentati nello schema di Figura 5-6 è evidente che la digitazione di un tasto collega una linea di uscita della Porta B con una di ingresso della Porta A. Essendo la linea PA collegata con 1 logico ne deriva che uno stato logico 1 su una qualsiasi delle linee di ingresso PA0 (PA0-PA3) identifica l'avvenuta digitazione di un tasto. CHECKB utilizza la tecnica "shot-gun" ("a mitraglia") inviando in uscita su PB1-PB7 altrettanti bit e determinando lo stato logico 1 di tutte le linee orizzontali di Figura 5-21 attigue ai tasti. CHECKB rileva la presenza di un tasto premuto mediante l'esecuzione di un'istruzione di ingresso della Porta 04 [IN A,(04H)] e l'ispezione dei quattro bit di ingresso per rilevare *almeno* un 1 logico. Vale la pena di ricordare che la porta A del PIO N. 1, è stata definita con funzionamento in Control Mode (Modo 3), cioè con bit di I/O predisposti secondo la configurazione 10001111, cosicchè PA0-PA3 corrispondono ad altrettanti bit di ingresso debitamente sottoposti a latch del PIO N. 1, i cui codici di dispositivo sono Porta 04 per i dati, e Porta 06, per il controllo. Purtroppo le informazioni sin qui ottenute non sono ancora sufficienti a determinare quale sia esattamente il tasto premuto tra i sette della colonna e perciò, per completare l'opera, è necessario l'intervento della routine KBSCAN. Possiamo per il momento osservare che CHECKB è una routine veloce e semplice che assolve egregiamente al suo compito. L'esecuzione della routine CHECKB viene ripetuta parecchie volte, con una percentuale relativamente bassa di controllo con risultati positivi, durante il periodo di tempo che trascorre in attesa della digitazione di un tasto. E' dunque più che opportuno mantenerla semplice. CHECKB è riprodotta nella sezione che segue, dedicata alla "Documentazione del Software di sistema", nell'esatta forma in cui essa compare nel sistema operativo del Nanocomputer.

DOCUMENTAZIONE DEL SOFTWARE DI SISTEMA
ROUTINE DI INGRESSO DA TASTIERA: CHECKB, IO E KBSCAN

SUBROUTINE CHECKB

Funzione

Stabilire se è stato premuto un tasto della tastiera. Il flag di zero è resettato in caso positivo, oppure settato in caso contrario.

Locazione di memoria

CHECKB (Per l'indirizzo assoluto della locazione si veda l'Appendice A - Tabella A-1).

Dati in ingresso

Nessuno.

Dati in uscita

Bit 0: è uguale a 1 se la colonna 1 contiene un tasto premuto altrimenti 0

Bit 1: è uguale a 1 se la colonna 2 contiene un tasto premuto, altrimenti 0

Bit 2: è uguale a 1 se la colonna 3 contiene un tasto premuto, altrimenti 0

Bit 3: è uguale a 1 se la colonna 4 contiene un tasto premuto, altrimenti 0

Il flag di zero è resettato se qualche tasto è premuto, e resettato in caso contrario.

La Tavola 5-2 riporta uno schema della disposizione della tastiera del Nanocomputer per righe e colonne.

Tavola 5-2. Layout della tastiera del Nanocomputer.

	Col. 1	Col. 2	Col. 3	Col. 4
Riga 1	0	1	2	3
Riga 2	4	5	6	7
Riga 3	8	9	A	B
Riga 4	C	D	E	F
Riga 5	→	←	ST	LA
Riga 6	2ND	SS	INC	LD
Riga 7	ARS	GO	RBK	DP

Registri Utilizzati

A

Descrizione

Il primo passo consiste nel portare allo stato alto tutte le linee orizzontali (si veda la Figura 5-6) della matrice della tastiera e disattivare il decodificatore/latch. Questo si ottiene ponendo in uscita alla porta 05, il valore 01 esadecimale, corrispondente alla porta B dei dati del PIO N. 1. In tal modo la pressione eseguita su di un qualsiasi tasto eseguirà un contatto con un 1 logico del collettore di uno dei transistori di Q1 (BGY16) e, di conseguenza, farà comparire un 1 logico sulla linea di ingresso corrispondente alla colonna del tasto premuto della porta 04 (porta A dei dati del PIO N. 1).

Trascorso l'intervallo di delay di 17 microsecondi, il contenuto della porta 04 è posto nell'accumulatore e sottoposto a AND con 0F esadecimale per fare assumere il corretto valore al flag di zero. Il controllo è poi restituito alla routine principale.

SUBROUTINE IO

Funzione

Inviare un byte in uscita verso i transistori di Q1 (BGY16) del circuito della tastiera allo scopo di accertare la presenza di eventuali tasti premuti.

Locazione in memoria

IO (Contenuta interamente all'interno della subroutine CHECKB, per l'indirizzo assoluto si veda l'Appendice).

Dati in ingresso

L'accumulatore contiene il byte da inviare in uscita ai transistori di Q1 (BGY16).

Dati in uscita

I quattro bit meno significativi dell'accumulatore si presentano come segue:

Bit 0: è uguale a 1 se la colonna 1 presenta un tasto premuto in una riga il cui transistore associato è in conduzione. In caso contrario 0.

Bit 1: è uguale a 1 se la colonna 2 presenta un tasto premuto in una riga il cui transistore associato è in conduzione. In caso contrario 0.

Bit 2: è uguale a 1 se la colonna 3 presenta un tasto premuto in una riga il cui transistore associato è in conduzione. In caso contrario 0.

Bit 3: è uguale a 1 se la colonna 4 presenta un tasto premuto in una riga il cui transistore associato è in conduzione. In caso contrario 0.

Il flag di zero è resettato se è stato rilevato qualche tasto premuto (accumulatore diverso da zero), e settato nel caso contrario.

Registri utilizzati

A, F

Descrizione

Prima di chiamare la subroutine IO, l'accumulatore è caricato con le informazioni relative alle righe da sottoporre a controllo per la ricerca del tasto premuto. Uno zero logico del bit Dn determina l'esame della riga n, un uno logico nel bit Dn permette che un tasto premuto nella riga n passi inosservato, dove n = 1, . . ., 7. (Si noti che il bit D0 è utilizzato per il segnale di strobe del decodificatore/latch ed è quindi irrilevante per questa routine). I valori logici dei bit dell'accumulatore da D1 a D7 sono dotati del significato di cui sopra per le seguenti ragioni:

- Le linee di ingresso alla base dei transistori di Q1 (BGY16) coincidono con i bit PB1-PB7 della porta 05 di uscita (corrispondente al bus dei dati della Porta B per quanto riguarda il PIO N. 1).
- Un'istruzione OUT (05H),A pone ciascuna di queste linee ai valori logici stabiliti dal contenuto dell'accumulatore.
- Uno zero logico applicato alla base di uno qualsiasi di questi transistori lo pone in conduzione, con la conseguente scomparsa dell'uno logico nel suo collettore.
- Quando un tasto è premuto un uno logico, in corrispondenza del collettore di un transistore, è trasferito ad uno dei quattro bit di ingresso della porta 04 di ingresso (corrispondente per il PIO N. 1 al bus dei dati della Porta A0. I bit settati alla porta 04 corrispondono alle colonne del generico tasto premuto.

- Perciò un'istruzione IN A,(04H) legge i quattro bit di stato delle colonne e li riporta nell'accumulatore onde poter verificare la presenza di un bit uguale a uno, che indicherebbe l'avvenuta digitazione di un tasto.

In base all'esposizione appena fatta dovrebbe essere facile comprendere le istruzioni costituenti la subroutine IO.

Listing completo di commenti - CHECKB e IO

```
CHECKB.  LD A,01H      ; Il registro A corrisponde al byte in uscita. Si noti che D0 è
                       ; alto, causando perciò l'inibizione dei display.
         EX (SP),HL    ; Ritardo di 17 microsecondi
         EX (SP),HL
    IO:  OUT (05H),A   ; PB1-PB7 sono posti a 0 logico, permettendo in tal modo che
                       ; tutte le linee orizzontali della tastiera passino allo stato alto.
         EX (SP),HL    ; Ritardo di 17 microsecondi
         EX (SP),HL
         IN A,(04H)    ; Legge le quattro linee di ingresso PA0-PA3
         AND 0FH       ; Se è stato premuto qualche tasto, il flag Z è resettato, in caso
                       ; contrario è settato
         RET           ; Ritorna al programma principale
```

I ritardi prima e dopo l'istruzione OUT (05H),A servono ad eliminare il rimbalzo del tasto, di cui si è detto in precedenza, e permettono la scarica dei transistori.

SUBROUTINE KBSCAN

Funzione

Scandisce la tastiera alla ricerca dei tasti premuti, rileva l'esistenza di un tasto premuto (uno solo) e ne comunica l'identità alla routine principale.

Locazione di memoria

KBSCAN (Per l'indirizzo della locazione si veda l'Appendice A - Tabella A-1).

Dati in ingresso

Nessuno

Dati in uscita

Il flag di carry è settato se i tasti premuti sono più di uno oppure nessuno. E' resettato se è stato premuto esattamente un solo tasto. Il numero del tasto premuto è posto nei registri A e C.

Registri utilizzati

A, C, F e B

Descrizione

Dopo l'inizializzazione dei registri B e C, in modo che: il registro B possa essere opportunamente incrementato per identificare un unico tasto premuto (se mai esso esiste) e il registro C punti alla riga 1 (associato al bit D0), si controlla ciascuna riga

per ricercare un eventuale tasto premuto. Questo esame utilizza la subroutine IO, con l'accumulatore caricato del complemento del contenuto del registro C. Se non si scopre alcun tasto premuto, il bit settato nel registro C viene posto, mediante rotazione, nel flag di carry, e viene eseguito il ritorno al programma principale (RET C) con il flag di carry settato. Se un ritorno dalla routine IO avviene con il flag di carry resettato, viene eseguito un loop nel corso del quale l'informazione contenuta nello accumulatore è utilizzata per incrementare il registro B esattamente di tanto quanto basti perchè il suo contenuto rifletta il numero della colonna di appartenenza del tasto premuto. Il bit settato più a destra dell'accumulatore corrisponde appunto alla colonna dove è stato premuto un tasto. Nel caso in cui i tasti premuti siano più d'uno dopo il trasferimento nel flag del carry del primo bit uno più a destra, viene effettuato un esame che controlla l'eventuale presenza di un bit settato tra i bit che vanno da 0 a 4 dell'accumulatore. Questo esame è realizzato utilizzando un'operazione di AND logico tra il contenuto dell'accumulatore ruotato ed il byte 0F. Se la routine IO ritorna al programma principale con uno solo dei bit 0-3 dell'accumulatore settato il risultato di questa operazione di AND è zero. Nel caso che l'AND dia luogo ad un risultato diverso da zero vi è stato più di un tasto premuto ed il controllo ritorna al programma principale con il flag di carry settato.
Nel caso che il risultato dell'AND sia zero, non vi è stato nessun altro tasto premuto nella riga associata al bit settato nel registro C. Il successivo gruppo di istruzioni, sino a LPKB1, provvede a controllare se un tasto è stato premuto in una delle altre righe. Questo è effettuato inviando in uscita alla porta 04 il valore del registro C (in luogo del suo complemento logico). Eseguendo le istruzioni IN A,(04H) e AND 0FH si ispezionano tutte le altre righe, diverse da quella puntata dal bit settato nel registro C, per ricercare un tasto premuto. Si osservi che, per tenere conto del rimbalzo dei tasti e del tempo di commutazione dei transistori è stato inserito un ritardo di 30 microsecondi. Anche in questo caso un risultato, dell'operazione AND, diverso da zero determina un ritorno al programma principale con il flag di carry settato. Se nelle altre righe non è stato premuto alcun tasto, quello premuto è l'unico e la sua riga è nota. Il registro B contiene un numero che, incrementato di 4(n) volta, per un tasto premuto nella riga n, fornisce il numero intero compreso tra 1 e 28 corrispondente al tasto. La funzione del loop LPKB1 è appunto quella di incrementare il numero memorizzato nel registro B. Il numero del tasto che ne risulta è quindi caricato dal registro A nel registro C. Le istruzioni CCF e RET restituiscono il controllo al programma principale con il numero corrispondente al tasto caricato nei registri A e C ed il flag di carry resettato.

Subroutine KBSCAN — Listing completo di commenti

```
KBSCAN:   LD BC,F701H    ; Inizializza B a F7 e C a 01
                         ; C è il puntatore alla riga di tasti in esame. B è a disposizione
                         ; come puntatore ad un eventuale tasto premuto
  LPKBS:  SLA C          ; Sposta C verso sinistra, con riempimento di zero per punta-
                         ; re alla prossima riga da esaminare
          RET C          ; Ritorna se il puntatore di riga, il bit 1, è stato ruotato nel
                         ; flag di carry, se cioè, la rassegna di tutte le righe è completa
          LD A,C         ; Muove il puntatore di riga nell'accumulatore, in quanto
                         ; l'uscita verso PB1-PB7 proviene da A
          CPL            ; Complementa l'accumulatore per invertire i transistori di
                         ; Q1
          CALL IO        ; Esamina la riga puntata dall'unico bit 0 in accumulatore
          JR Z,LBPKBS    ; Se la subroutine IO setta il flag Z, nella riga esaminata non
                         ; vi sono tasti premuti. Prova quindi la prossima riga
  STPKB:  INC B          ; Se la subroutine IO resetta il flag Z, vi è un tasto premuto.
                         ; Di quale tasto si tratta? Incrementa il registro B del numero
                         ; di posti di cui l'accumulatore è stato ruotato verso sinistra.
```

```
         RRA           ; Ruota verso destra l'accumulatore sino a trovare un bit
                       ; settato contando per mezzo di incrementi del registro B
         JR NC,STPKB   ; Continua a ruotare sino a che il flag di carry non viene
                       ; settato
         AND 0FH       ; Una volta settato il flag di carry, controlla se vi sono ulte-
                       ; riori bit a uno. Se il tasto premuto è unico la operazione di
                       ; AND dovrebbe avere come risultato 00
         JR NZ,CCF     ; Se viene premuto più di un tasto provoca un ritorno con il
                       ; flag di carry settato (CCF significa "complementa il flag di
                       ; carry", per cui esso risulta settato in quanto l'istruzione
                       ; AND lo aveva resettato)
         LD A,C        ; Trasferisci il contenuto del registro C (il puntatore della
                       ; riga in cui si trova il tasto premuto) nell'accumulatore
         INC A         ; Setta il bit D0 per attivare il pin di inibizione del dispositi-
                       ; vo decodificatore
         CALL IO       ; Esamina tutte le altre righe alla ricerca di un tasto premuto
         EX (SP),HL    ; Ritardo di 30 microsecondi
         EX (SP),HL
         EX (SP),HL
         EX (SP),HL
         IN A,(04H)    ; Con questa istruzione non vi dovrebbero essere bit di ingres-
                       ; so settati
         AND 0FH       ; Esamina se ci sono dei bit uno
         JR NZ,CCF     ; Ritorna con il flag di carry settato nel caso che vi sia qual-
                       ; che tasto premuto in una riga diversa da quella puntata dal
                       ; registro C
         LD A,B        ; Il registro B è stato aggiornato con il valore corrispondente
                       ; alla colonna del tasto premuto (si veda il loop STPKB), poi
                       ; aggiungendo 4 per ogni zero che si trova alla destra del bit
                       ; a uno nel registro C aggiorna B per individuare la riga cor-
                       ; rispondente al tasto premuto. Per effettuare questa opera-
                       ; zione di incremento B è caricato in A
LPKB1:   ADD, A,04H    ; B è stato inizializzato in modo che debba almeno una volta
                       ; essere addizionato 4
         SRL C         ; Controlla il puntatore di riga (bit settato)
         JR NC,LPKB1   ; Somma un altro 4 se il bit è resettato
         LD C,A        ; Una volta che il bit settato è stato ruotato nel flag di carry,
                       ; l'accumulatore contiene il numero (0-27) del tasto premu-
                       ; to. Carica tale numero nel registro C
CCF:     CCF           ; Complementa il flag di carry. Se LPKB1 è stato appena ese-
                       ; guito, il flag di carry è stato settato per oltrepassare l'istru-
                       ; zione JR NC cosicchè CCF resetta il flag di carry.
         RET           ; Ritorna con il flag di carry settato se non è stato premuto
                       ; nessuno o più di un tasto con il flag di carry resettato ed il
                       ; numero del tasto nei registri A e C se è stato premuto solo
                       ; un tasto.
```

ROUTINE DI VISUALIZZAZIONE: CONVDI E DISPL

Le due routine che seguono, CONVDI e DISPL, sono utilizzate dal sistema operativo del Nanococmputer per la gestione dei due blocchi di sette led e degli otto display a sette segmenti. CONVDI provvede alla lettura di un insieme prestabilito di dieci locazioni di memoria (una per ciascuna unità del display) e traduce il loro contenuto in forma idonea a far accendere i led e/o i segmenti appropriati, ponendola in uscita alla Porta B dei dati del PIO N. 1. DISPL è adibita al compito vero e proprio di uscita,

provvedendo a leggere i byte tradotti, generati da CONVDI, per porli quindi in uscita alla Porta B dei dati del PIO N. 1. Una tipica serie di passi, di cui si vale il sistema operativo del Nanococmputer per gestire le dieci unità di display, si presenta come segue:

1. Poni i dati da visualizzare nelle locazioni che vanno da LEDH a ADD7 +7 (per maggiori particolari si veda in Appendice la sezione sullo spazio della RAM di lavoro del Nanocomputer nonchè la documentazione su CONVDI e DISPL)
2. Chiamata di CONVDI per effettuare la traduzione
3. Un loop interno alla subroutine chiama DISPL sintantochè non viene rilevato un ingresso proveniente dalla tastiera.

Il loop del Passo 3 è essenziale alla rigenerazione di tutti e dieci i display in modo che questi si presentino, all'occhio umano, in modo stabile.

SUBROUTINE CONVDI

Funzione

Tradurre una stringa di quattro byte binari negli otto codici esadecimali a sette segmenti, a otto bit, ad essi associati, idonei ad essere visualizzati sull'unità di display del Nanocomputer.

Locazione in memoria

CONVDI (Per la locazione assoluta vedasi Appendica A, Tabella A-1).

Dati in ingresso

I quattro byte in ingresso (da tradurre in otto digit esadecimali) sono memorizzati in questo modo:

DATAL – Byte di dati LO
DATAH – Byte di dati HI
ADDL – Byte d'indirizzo LO
ADDH – Byte d'indirizzzo HI
(Per l'indirizzo assoluto delle locazioni si veda l'Appendice).

Il registro accumulatore alternativo (A') specifica quale display a sette segmenti debba essere acceso o spento in base alla convenzione seguente:

Bit 0 – un uno logico implica che il digit esadecimale meno significativo del display dati debba essere spento; uno zero logico implica che il digit esadecimale specificato (i quattro bit meno significativi di DATAL) sia visualizzato

I bit da 1 a 7 hanno significato analogo

La coppia di registri DE = ADDH (ultimo da tradurre nella stringa di 4 byte)
La coppia dei registri HL = ADD7 – 1 (indica la locazione precedente a quella del primo digit di destra della stringa di 8 da visualizzare)

Dati in uscita

I codici a sette segmenti per le otto cifre esadecimali del display sono memorizzati come segue nelle locazioni da ADD7 a ADD7 + 7:

ADD7 – codice esadecimale per il display a sette segmenti, per i quattro bit più significativi del byte d'indirizzo HI.

ADD7 + 1 – codice esadecimale per il display a sette segmenti, per i quattro bit meno significativi del byte d'indirizzo HI.

ADD7 + 2 – codice esadecimale per il display a sette segmenti, per i quattro bit più significativi del byte d'indirizzo LO.

ADD7 + 3 – codice esadecimale per il display a sette segmenti, per i quattro bit meno significativi d'indirizzo LO.

DATA7 – ADD7 + 4 – codice esadecimale per il display a sette segmenti, per i quattro bit più significativi del byte di dati HI.

DATA7 + 1 = ADD7 + 5 – codice esadecimale per il display a sette segmenti, per i quattro bit meno significativi del byte di dati HI.

DATA7 + 2 = ADD7 + 6 – codice esadecimale per il display a sette segmenti, per i quattro bit più significativi del byte di dati LO.

DATA7 + 3 = ADD7 + 7 – codice esadecimale per il display a sette segmenti, per i quattro bit meno significativi del byte di dati LO.

Registri utilizzati

TUTTI (La coppia di registri DE è conservata nello stesso stato che aveva all'ingresso della routine).

Descrizione

Le prime tre istruzioni salvano il contenuto della coppia di registri DE (puntatore di ingresso) nello stack, scambiano i contenuti delle coppie di registri DE ed HL (così che il puntatore dell'ingresso si trova da questo momento in HL) ed azzerano l'accumulatore. Queste due ultime disposizioni iniziali saranno usate nel seguito dall'istruzione RDL, per la rotazione di successive sezioni di 4 bit nell'accumulatore, per la traduzione negli equivalenti codici esadecimali a sette segmenti.

La logica della subroutine CONVDI è basata su di un loop principale, LOOPXX, eseguito per otto volte man mano che la coppia DE è incrementata progressivamente da LEDL a ADD7 + 7. La istruzione INC DE è eseguita prima del test finale che controlla il completamento della conversione di tutte le otto sezioni. DE comincia quindi a LEDL ed il test finale confronta DE con ADD7 + 8.

Il loop LOOPXX fa ruotare nei quattro bit meno significativi dell'accumulatore un gruppo di quattro bit della stringa in ingresso di quattro byte. L'esecuzione di LOOPXX è ripetuta per quattro volte, utilizzando il registro B come contatore, per effettuare uno scorrimento a sinistra (riempimento di zeri del contenuto delle locazioni di memoria DATAL-ADDH). Per la rotazione viene seguito il seguente ordine: dal gruppo più significativo a quello meno significativo, cioè dal gruppo più alto di ADDH a quello più basso di DATAL. Nella Figura 5-21 è riportato il diagramma che rappresenta questa operazione.

Una volta che un gruppo si trova al posto dei quattro bit meno significativi dello accumulatore, si procede alla sua conversione nell'equivalente esadecimale per il display a sette segmenti. Il valore binario rappresentato dai quattro bit è utilizzato per la gestione di una tabella chiamata SEGTAB, riprodotta nella pagina successiva.

Locazione	Contenuto	Segmenti accesi	Cifra esadecimale
SEGTAB	FC	a,b,c,d,e,f	0
	60	b,c	1
	DA	a,b,d,e,g	2
	F2	a,b,c,d,g	3
	66	b,c,f,g	4
	B6	a,c,d,f,g	5
	BE	a,c,d,e,f,g	6
	E0	a,b,c	7
	FE	a,b,c,d,e,f,g	8
	F6	a,b,c,f,g	9
	EE	a,b,c,e,f,g	A
	3E	c,d,e,f,g	B
	9C	a,d,e,f	C
	7A	b,c,d,e,g	D
	9E	a,d,e,f,g	E
	8E	a,e,f,g	F
	00	(Nessun segmento)	Vuoto

Si osservi che la corrispondenza tra i segmenti del display a sette segmenti ed i bit di un byte è la seguente:

Segmento	Bit
a	7
b	6
c	5
d	4
e	3
f	2
g	1

Il bit zero non è utilizzato. Un segmento acceso corrispondente ad un 1 logico, un segmento spento ad uno 0 logico.
Una volta che si sia determinato il corretto codice esadecimale a sette segmenti, il registro A' é ruotato di un posto per porre nel flag di carry il prossimo bit più significativo. La presenza di un carry azzera il codice esadecimale; in caso contrario il codice esadecimale viene trascritto nell'apposito byte di destinazione.

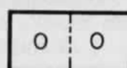
LOOPXX viene ripetuto sino a che non siano stati tradotti tutti e quattro i byte. Prima che il controllo sia restituito al programma principale si procede al ripristino del contenuto primitivo della coppia di registri DE.

Osservazione finale. La stringa originale di quattro byte in ingresso è opportunamente salvata mediante queste istruzioni:

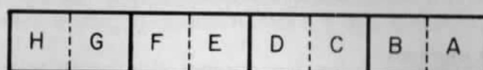
```
PUSH AF         ; Per salvare il gruppo fatto ruotare nella metà meno signifi-
                ; cativa del registro A
POP AF          ; Per ripristinare il registro A
LD HL,DATAL     ; Per porre quello che era il gruppo più significativo nel loop
ADD A,(HL)      ; precedente al posto del gruppo meno significativo del loop
LD (HL),A       ; successivo
```

Da queste istruzioni si ottiene praticamente una *"circolazione"* dell'istruzione RDL, con l'effetto che il gruppo più significativo esce per rotazione dall'estremità sinistra della stringa per rientrare da quella destra.

Valori iniziali:

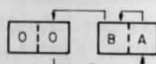


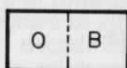
Accumulatore



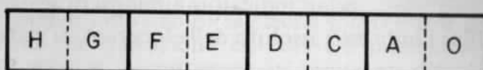
ADDH ADDL DATAH DATAL

Dopo il primo RLD:



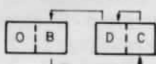


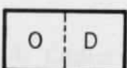
Accumulatore



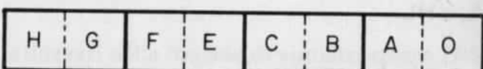
ADDH ADDL DATAH DATAL

Dopo il secondo RLD:



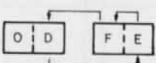


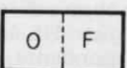
Accumulatore



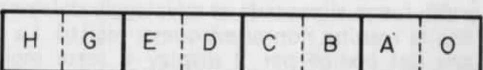
ADDH ADDL DATAH DATAL

Dopo il terzo RLD:



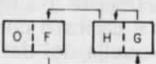


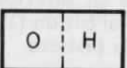
Accumulatore



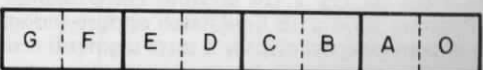
ADDH ADDL DATAH DATAL

Dopo il quarto RLD:





Accumulatore



ADDH ADDL DATAH DATAL

Figura 5-21. Operazione di scorrimento decimale a sinistra e riempimento di zeri su due cifre del display dati e indirizzi.

SUBROUTINE DISPL

Funzione

Visualizza l'indirizzo, il dato e l'informazione relativa al selettore memorizzato alle locazioni ADD7; DATA7 e LEDH.

Locazione di memoria

DISPL (Per la locazione assoluta vedasi Appendice A - Tabella A-1).

Dati in ingresso

ADD7 — indirizzo di partenza per i quattro codici di un byte necessari a pilotare i quattro display a sette segmenti nel display degli indirizzi

DATA7 — indirizzo di partenza per i quattro codici di un byte necessari a pilotare i quattro display a sette segmenti nel display dei dati

LEDH — indirizzo di partenza per i due byte i cui bit accendono gli appropriati indicatori luminosi di selezione (14 indicatori in tutto)

(Per l'indirizzo assoluto delle locazioni si veda l'Appendice)

Dati in uscita

Comando dei display.

Registri utilizzati

F, A, e HL.

Tempo totale di esecuzione

940 microsecondi.

Descrizione

In primo luogo si procede al salvataggio del contenuto della coppia di registri BC; in modo che la routine non alteri questi registri. La coppia di registri HL è caricata con l'indirizzo del codice per il display a sette segmenti corrispondente al digit più significativo del display degli indirizzi. Ripetizioni successive del loop OUTLP comandano i rimanenti display. Il registro B, utilizzato per inviare il segnale di strobe al decodificatore che attiva selettivamente i singoli display, è inizializzato con il valore esadecimale 13. Il registro C è caricato con 05, corrispondente al codice del dispositivo di Porta 5. La Porta 5 è quella dei dati della Porta B del PIO N. 1: il bit D0 corrisponde al segnale di strobe inviato al decodificatore/latch HCF4514B, mentre i bit da D1 a D4 servono come segnali di ingresso di selezione inviati al decodificatore da 4 a 16 linee/latch oppure (congiuntamente ai bit da D5 a D7) sono adibiti al comando del display a sette segmenti e dei LED della tastiera.

Dopo aver posto in uscita, su ciascuna linea dei dati, degli uno logici per portare in interdizione i transistori di Q1 (BGY 16), che pilotano i display, la subroutine DISPL esegue un breve loop di ritardo, denominato LP.

Il digit o i gruppi di LED che devono essere visualizzati per primi sono preparati per la visualizzazione effettuando il caricamento del contenuto di HL nell'accumulatore, complementandolo e portando il bit 0 allo stato logico zero.

Fermiamoci ad esaminare la relazione tra l'hardware che compone il sistema di display della tastiera, ed il software, in particolare la subroutine DISPL, che interagisce con esso. La rappresentazione in memoria dei digit o dei LED da visualizzare si ottiene con la seguente convenzione: un bit allo stato logico uno corrisponde ad un segmento o a un LED acceso, il bit D0 rimane inutilizzato. Riferendosi al diagramma del display della tastiera, si noti come una linea dei dati allo stato logico 0 accenda un segmento un LED previa polarizzazione del relativo transistore (il collettore si porta allo stato logico 1, i diodi del display risultano in tal modo polarizzati diretta-

mente e possono così accendersi). Il decodificatore HCF4514B, Q4 nello schema, seleziona il display ponendo su una linea di uscita un uno logico che porta in conduzione un transistore, abbassando quindi il potenziale del suo collettore al quale fanno capo i diodi del display selezionato. Il decodificatore è attivo con il bit D0 allo stato logico 0.

L'ultimo paragrafo dovrebbe aver chiarito la ragione per cui il software DISPL complementa l'accumulatore e resetta il bit 0 come passo preliminare all'invio (in uscita) di un byte di pilotaggio del display. Le tre istruzioni:

```
OUT  (C),B
DEC  B
OUT  (C),B
```

creano il segnale di strobe per un nuovo indirizzo di selezione (bit D1-D4) che deve entrare nel decodificatore mediante la transizione del bit D0 basso-alto-basso. Effettuata la selezione del giusto display, viene posto in uscita un digit, oppure un byte che definisce il gruppo di LED, mediante l'istruzione OUT (PORT5),A. Il registro B comincia dal valore 13 esadecimale, cosicchè ogni doppio decremento per ogni byte visualizzato permette il caricamento degli indirizzi appropriati nel decodificatore/latch. Nel registro B, al completamento della rassegna di tutti i display è presente il valore esadecimale FF. La Tabella 5-4 riporta il codice esadecimale per

Tabella 5-4. Codici esadecimali utilizzati nella routine di visualizzazione DISPL.

Lettera	Codice Esadecimale	Lettera	Codice Esadecimale
A	EE	t	IE
b	3E	U	7C
C	9C	u	38
D	F0	v	38
d	7A	W	7C, 70
E	9E	w	38, 30
F	8E	Y	4E
G	BC	Z	DA
H	6E	=	12
h	2E	–	02
I	60	—	10
i	20	°	C6
J	F0	?	CA
K	4E	\	4A
L	1C	/	26
M	EC, E0	1	0C
m	2A, 22	2	DA
N	EC	3	F2
n	2A	4	66
O	FC	5	B6
o	3A	6	BE
P	CE	7	EO
R	AC	8	FE
r	0A	9	F6
S	B6	0	FC

Si noti che per M e W i byte sono due.

il display a 7 segmenti, ciascuna lettera dell'alfabeto, ai numeri decimali o a varie specie di simboli. Questi codici possono essere usati, insieme alla subroutine DISPL, per creare messaggi analoghi a quello visualizzato dalla routine di caricamento dal software per esperimenti posta in F000 esadecimale.

Dopo che ciascuno dei byte è stato inviato in uscita viene eseguita una attesa, con il loop AAAA. L'istruzione successiva ad AAAA gestisce la logica necessaria per (a) determinare quando sono stati visualizzati i digit dell'indirizzo (perchè possa avere inizio la visualizzazione dei digit dei dati); (b) determinare quando è terminata la visualizzazione di tutti i digit del dato (in modo che possa avere inizio la visualizzazione degli indicatori di selezione) e (c) se tutto il lavoro di visualizzazione è stato portato a termine (DEC B, JP,OUTLP).

Con l'ultima istruzione OUT (C),B su tutte le linee dei dati è imposto lo stato logico uno, per interdire tutti i transistori adibiti al pilotaggio del display e bloccare il decodificatore, in preparazione di nuove operazioni di I/O. Viene quindi ripristinata la coppia di registri BC nella sua configurazione originaria e viene ridato il controllo al programma principale.

Listing completo di commenti – CONVDI e DISPL

```
CONVDI:  PUSH DE        ; Salva il contenuto della coppia di registri DE, in modo da
                        ; poterlo ripristinare nel suo stato iniziale prima di tornare al
                        ; programma principale.
         EX DE,HL       ; Scambia DE con HL in modo che DE punti alla prima loca-
                        ; zione che precede quella del digit più a destra nella stringa
                        ; degli 8 digit convertiti per la visualizzazione.
         XOR A          ; Azzera l'accumulatore in previsione dell'istruzione RLD
LOOPXX:  LD B,04H       ; Il registro B è adibito al conteggio del numero di RLD ese-
                        ; guiti per la rotazione della stringa di quattro byte (si veda
                        ; DJNZ LOOPX).
         INC DE         ; Aggiorna DE in modo che punti alla prima locazione desti-
                        ; nata alla memorizzazione dei digit convertiti.
         LD HL,DATAL    ; HL punta adesso alla locazione 0FE2, la meno significativa
                        ; per la stringa di 4 byte fatta ruotare. Questa stringa di 4
                        ; byte è soggetta ad operazioni RLD mano a mano che
                        ; ciascun gruppo successivo di 4 digit è ruotato nell'accumu-
                        ; latore, convertito e memorizzato in (DE).
         ADD A,(HL)     ; "Circola" l'operazione RLD addizionando all'accumulatore,
                        ; il cui gruppo di 4 bit di sinistra è 0, il byte meno significati-
                        ; vo della stringa di 4 byte fatta ruotare, il cui gruppo di 4 bit
                        ; di destra è stato riempito di zeri. Ne risulta che il gruppo di
                        ; destra nell'accumulatore (che ERA il gruppo più significati-
                        ; vo della stringa di 4 byte prima dell'ultimo RLD) diventa il
                        ; gruppo meno significativo. Si veda l'illustrazione allegata
                        ; alla documentazione su CONVDI.
         LD (HL),A      ; Aggiorna il byte meno significativo
         LD A,E         ; Controlla se l'ultimo gruppo è stato tradotto verificando che
         CP 0C2H        ; DE punti all'ultimo byte della stringa di 8 byte da con-
                        ; vertitre
         JR Z,KBINI     ; In caso affermativo salta all'istruzione di ripristino di DE
                        ; e RET
         XOR A          ; Reinizializza A prima di procedere allo scorrimento di 4
                        ; byte con l'istruzione RLD
LOOPX:   RLD            ; Rotazione sinistra decimale
         INC HL         ; Punta al byte che segue in ordine di importanza
         DJNZ LOOPX     ; Ruota 4 volte
```

```
         PUSH AF            ; Salva l'accumulatore
         LD HL,SEGTAB       ; Traduce il gruppo di 4 bit contenuto nella metà inferiore
         ADD A,L            ; dell'accumulatore utilizzando quest'ultimo come indice
         LD L,A             ; nella tabella per i display
         LD A,(HL)          ; a sette segmenti
         LD C,A             ; Salva il gruppo appena convertito nel registro C
         EX AF,A*F'         ; Giudica se questo byte dovrebbe essere visualizzato osser-
                            ; vando il registro A'
         RLC A              ; Se il corrispondente bit di A' è settato il byte non deve
                            ; essere visualizzato e deve perciò essere azzerato
         JR NC,NOBLXX       ; Salta l'istruzione di azzeramento se il flag di carry è reset-
                            ; tato
         LD C,00H           ; Azzera C nel caso che il flag di carry sia settato
NOBLXX:  EX AF,AF'          ; Ripeti lo scambio al contrario
         LD A,C             ; Copia in A il gruppo appena tradotto
         LD (DE),A          ; Memorizza il byte tradotto per la sua successiva visualizza-
                            ; zione mediante le subroutine DISPL
         POP AF             ; Ripristina il gruppo di 4 bit nella forma analoga a quella
                            ; precedente alla conversione per preparare la "circolarizza-
                            ; zione" dell'operazione RLD
         JR LOOPXX          ; Torna indietro per cominciare a tradurre il prossimo byte
KBINI:   POP DE             ; Ripristina DE
         RET
```

```
DISPL:   PUSH BC            ; Salva il contenuto della coppia di registri BC nello stack in
                            ; modo da poterla ripristinare nello stato di partenza
         LD HL,DATA7-1      ; Carica la coppia di registri HL con l'indirizzo DATA7-1
                            ; corrispondente a 0FDB. DATA7-1 equivale a ADD7+3
                            ; che è l'indirizzo del digit più significativo del display degli
                            ; indirizzi (tradotto da CONVDI). Ne risulta che i quattro
                            ; digit dell'indirizzo saranno visualizzati per primi, con in
                            ; testa il byte più significativo
         LD BC,1330H+05H    ; Carica B con 13, C con 05. Si osservi che 05 è l'indirizzo di
                            ; porta della Porta B del PIO N. 1 adibita al pilotaggio dei
                            ; display
                            ; Il contenuto del registro B è inviato in uscita al decodifica-
                            ; tore per la selezione dell'unità di display che interessa.
```

CORRISPONDENZA TRA I VALORI DEL REGISTRO B, IL DISPLAY SELEZIONATO E L'INDIRIZZO DI MEMORIA VISUALIZZATO

Ad ogni esecuzione di OUTLP corrispondono due decrementi del registro B

Registro B	Display selezionato	Indirizzo di memoria visualizzato
11 o 10	ADD1	ADD7 + 2
0F o 0E	ADD2	ADD7 + 1
0D o 0C	ADD3	ADD7
13 o 12	ADD0	ADD7 + 3
0B o 0A	DATA0	DATA7 + 3
09 o 08	DATA1	DATA7 + 2
07 o 06	DATA2	DATA7 + 1
05 o 04	DATA3	DATA7
03 o 02	IY,IX,HL,DE,BC,AF,IR	LEDH + 1
01 o 00	ARS,ERR,SP,PC,MEM,I/O,BRK	LEDH

Dove ADD0-ADD3 costituiscono i quattro display degli indirizzi elencati da sinistra a destra

DATA0-DATA3 corrispondono ai quattro display dei dati, elencati da sinistra a destra

```
OUTLP:  LD A,FFH        ; Poichè il bit D0 —e alto, l'ingresso inhibit del latch/decodi-
                        ; ficatore HCF4514B risulta attivato
        LD A,03H        ; Loop di ritardo di 19,2 microsecondi
   LP:  DEC A
        JR NZ,LP
        LD A,(HL)       ; Carica in A il primo byte da visualizzare
        CPL             ; Complementa l'accumulatore per tenere conto dell'effetto
                        ; invertente dei transistori di Q1.
        RES 0,A         ; Resetta il bit D0 in modo che l'ingresso inhibit del latch/de-
                        ; codificatore cessi di essere attivo
        OUT (C),B       ; Aziona il byte di selezione del display presente nel registro
                        ; B, introducendolo nel latch del dispositivo latch/decodi-
                        ; ficatore
        DEC B           ; Primo decremento del registro B
        OUT (C),B
        OUT (05H),A     ; Byte da visualizzare inviato in uscita verso PB1-PB7
        LD A,10H        ; Altro breve loop di ritardo
 AAAA:  DEC A
        JR NZ,AAAA
        DEC HL          ; Aggiorna HL in modo da puntare al prossimo byte da
                        ; visualizzare
        LD A,L          ; Verifica che tutti i byte ADDR
        CP 0B9H         ; siano stati visualizzati
        JR NZ,NXT1      ; In caso negativo torna indietro per visualizzare il byte ADDR
                        ; successivo
        LD L,0C1H       ; In caso affermativo, inizializza L per la visualizzazione dei
                        ; byte DATA
 NXT1:  CP 0BDH         ; Verifica che tutti i byte DATA
        JR NZ,NXT2      ; siano stati visualizzati
        LD L,0B9H       ; In caso affermativo inizializza L per il display dei byte LED
 NXT2:  DEC B           ; In caso negativo prosegui. Secondo decremento del registro
        JP P,OUTLP      ; B fintantochè B è zero, o maggiore di zero, continua nella
                        ; visualizzazione
        OUT (C),B       ; Non appena B diventa negativo, cioè uguale ad FF, invia in
                        ; uscita un D0 alto per l'inibizione del latch/decodificatore
        POP BC          ; Ripristina la coppia di registri BC
        RET
```

Routine per I/O seriale

Nei casi sin qui trattati, tutti i circuiti di I/O del microcomputer che abbiamo presentato, scambiano byte di dati con la CPU Z80 mediante trasmissione di bit in *parallelo.* Questa equivale alla trasmissione di otto bit, costituenti una comunicazione verso (oppure dalla) la CPU, su di otto apposite linee distinte, ossia con una linea riservata a ciascun bit. Questo tipo di comunicazione, perfettamente accettabile per dispositivi tra di loro molto vicini, diventa economicamente insostenibile per dispositivi distanti dalla CPU diverse decine di metri, o addirittura chilometri, a causa del costo di installazione delle otto linee necessarie a sostenere l'I/O parallelo. Perciò,

sulle lunghe distanze, l'I/O seriale si pone come alternativa, preferibile in termini economici, rispetto alla comunicazione in parallelo. Le interfacce seriali con la CPU Z80 assolvono alle funzioni seguenti:

Ingresso seriale:
1. Ricevere dalla periferica i dati in serie, ad un bit per volta.
2. Ricostituire il byte di otto bit
3. Porre il byte di otto bit nel registro interno della CPU espressamente indicato dal software come registro di destinazione.

Uscita seriale:
1. Ricevere il byte di otto bit da inviare in uscita alla periferica
2. "Serializzare" il byte, ossia inviare in uscita verso la periferica il byte, ad un bit per volta.

In entrambi i casi la CPU Z80 comunica con i circuiti adibiti all'I/O in serie mediante byte di otto bit *paralleli*. La funzione dell'interfaccia seriale è appunto quella di operare la trasformazione da serie a parallelo oppure da parallelo a serie e di provvedere alla temporizzazione della scrittura o lettura dei dati sul mezzo fisico di comunicazione.

Nei circuiti per I/O parallelo, già presentati nel Capitolo 4, per mantenere il coordinamento tra l'operazione di scrittura, da parte del dispositivo trasmettitore, e quello di lettura, da parte dell'unità ricevente, sul mezzo usato per la comunicazione (D0-D7), abbiamo fatto ricorso ad impulsi di sincronizzazione quali IN XX e OUT XX. Appunto a questa funzione erano adibite delle linee distinte, dette *linee di controllo*. Per la comunicazione seriale è altresì richiesta una condizione di coordinamento analoga, con la differenza che non vi sono a disposizione linee extra da adibire a tale funzione. (Si osservi, comunque, che alcuni mezzi per comunicazione seriale sono costituiti da un numero di fili superiore a quello strettamente necessario cioè: ingresso, uscita, massa e alimentazione). Per il coordinamento delle attività dell'unità ricevente e di quella trasmittente si utilizzano le convenzioni che seguono:

1. Sia il ricevitore che il trasmettitore sono d'accordo nel campionare la linea secondo una certa frequenza di campionamenti al secondo. Tale frequenza di campionamento è detta BAUD rate (frequenza di baud) ed è contraddistinta dallo stesso valore per l'unità ricevente come per quella trasmittente.
2. L'unità ricevente e quella trasmittente hanno concordato preventivamente che ciascuno dei byte trasmessi sarà costituito da una sequenza di variazioni del livello di tensione sulla linea di trasmissione per il trasmettitore e la linea di ricezione per il ricevitore. Poichè sono collegati entrambi da una linea di massa, tutti e due gli interlocutori si varranno dello stesso punto di riferimento nella determinazione del livello della tensione di linea. Le conversioni mediante le quali, ai livelli di tensione, associano i livelli logici zero ed uno, sono di norma fissate nel seguente modo:

   Tensione $\leqslant V_L$ implica che il bit campioanto abbia valore logico 0

   Tensione $\geqslant V_H$ implica che il bit campionato abbia valore logico 1.

3. Sia l'unità ricevente che quella trasmittente hanno concordato che ciascuno dei byte (o successione di otto bit) deve essere preceduto da un bit basso, detto bit di START, che indica l'imminente trasmissione di un nuovo byte. Per indicare che la trasmissione del byte è terminata, devono essere trasmessi due bit alti, detti bit di STOP.

Le convenzioni 1, 2 e 3 possono presentare molteplici variazioni, soprattutto per quando riguarda le corrispondenze livelli logici-livelli di tensione ed il numero dei bit di start e stop. Le convenzioni qui esposte sono quelle utilizzate dall'interfaccia seriale del Nanocomputer.

La Figura 5-22 illustra il modo in cui si presenterebbero i livelli logici su di un mezzo adibito alla comunicazione seriale in caso di trasmissione del byte AC. Si

osservi come, dopo il completamento della trasmissione di un byte, lo stato di riposo della linea sia quello alto. Quando il livello di tensione si abbassa, questo va inteso come trasmissione di un bit di start premesso ai bit di un dato. Dopo la trasmissione di D7 la linea passa allo stato alto per un minimo di due intervalli di campionamento in quanto sono trasmessi i due bit di stop.

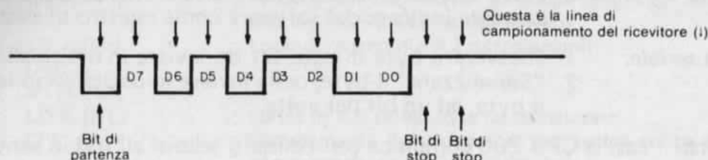


Figura 5-22. Trasmissione in serie di un carattere di otto bit con un bit di start e due bit di stop.

La funzione di trasformazione da parallelo a serie e da serie a parallelo, svolta da un'interfaccia seriale, può essere implementata sia in hardware che in software.
Svariati produttori di semiconduttori commercializzano dei dispositivi a circuiti integrati che eseguono la funzione serializzante/deserializzante insieme a molteplici altre funzioni aventi lo scopo di individuare e segnalare il verificarsi di errori di trasmissione. Questi IC sono denominati UART (Universal Asynchronous Receiver/Transmitter = ricevitore/trasmettitore asincrono universale) o anche USART (Universal Synchronous/Asynchronous Receiver/Transmitter = ricevitore/trasmettitore sincrono/asincrono universale). Mediante un normale kit di espansione è possibile dotare la scheda del Nanocomputer di un USART. Poichè nella configurazione standard del Nanocomputer non è previsto l'USART; la funzione di serializzazione/deserializzazione è realizzata via software dal sistema operativo mediante due subroutine, TTYI1, per l'ingresso seriale, e TTYO, per l'uscita seriale. Entrambe queste due routine utilizzano il bus dei dati della Porta A del PIO N. 1 per le combinazioni con i dispositivi seriali interfacciati con il Nanocomputer. Tramite il Connettore J3 viene interfacciato il registratore a cassette e al connettore J5 è possibile collegare periferiche seriali che operino in RS232-C, loop di corrente a 20 mA o con livelli TTL. Come abbiamo già spiegato, il PIO N. 1 è programmato dal sistema operativo in Control Mode, con i bit dei dati della Porta A nella seguente configurazione:

10001111

In altre parole, i bit PA7, PA3, PA2, PA1 e PA0 sono utilizzati per l'ingresso, mentre i bit PA6, PA5 e PA4 per l'uscita. La routine TTYI1 accetta dati seriali in ingresso dalla linea PA7, mentre la routine TTYO invia dati seriali in uscita sulla line PA4. Nello schema di principio relativo ai circuiti dell'unità tastiera/display, si osservi che PA4 è identificato con TXD (dati in trasmissione) e PA7 con RXD (dati in ricezione).

SUBROUTINE TTYI1

Funzione

Pone nei registri e nell'accumulatore un carattere di sette bit, trasmesso serialmente alla CPU. Si presuppone che le modalità di trasmissione del byte comportino un bit di start e due bit di stop.

Locazione di memoria

TTYI1 (Per l'indirizzo assoluto si veda l'Appendice A - Tabella A-1).

Dati in ingresso

Nessuno.

Dati in uscita

Il carattere ricevuto dalla CPU è caricato nei registri A e C.

Registri utilizzati

A, B e C.

Descrizione

Si attende, per prima cosa, l'ingresso di un bit di start (0 logico). Questa prima fase è realizzata dalle prime tre istruzioni del programma. Si effettua una lettura continua della porta di ingresso, codice di dispositivo 04, sintantochè D7 non si abbassi. Questa condizione è segnalata da un flag di carry resettato; infatti con l'istruzione RLA viene fatto ruotare D7 nel bit di carry.

Per accertarsi che è stato realmente posto un bit di start sulla linea, da parte di un dispositivo di trasmissione, si effettua la chiamata della routine BAUDHF.
Questa routine è costituita da un loop di ritardo di durata pari alla metà esatta di un periodo di campionamento. Dopo un tempo pari a un mezzo periodo di campionamento, si procede nuovamente al campionamento della porta 04 in modo da accertarsi che la linea D7 sia ancora bassa. Se D7 non si trova più nello stato logico 0, il breve segnale basso iniziale non è da considerarsi come un vero bit di start per cui il controllo è rinviato all'inizio della routine. Se, d'altra parte, la linea D7 è ancora bassa, il bit campionato è un vero bit di start. In questo caso ha inizio la fase di ingresso degli otto bit del dato.

L'istruzione LD B,09H inizializza il registro B come contatore dei bit. La ragione per cui il conteggio venga inizializzato a 9 anzichè a 8 dipende dal conteggio del primo bit campionato che è sempre il bit di start. Viene quindi eseguita per nove volte LOOPTI in modo da comprendere il bit di start e ciascuno degli otto bit del dato. Il primo bit ricevuto è posto nel bit D7 dell'accumulatore. L'istruzione RLA esegue la trascrizione del bit D7 nel flag di carry mentre l'istruzione RR C fa ruotare i bit del registro C, di una posizione verso destra, ponendo nel bit D7 il contenuto del flag di carry. I bit ricevuti vengono fatti ruotare ad uno ad uno, nel registro C. I bit meno significativi sono quelli che arrivano per primi e si trasferiscono nel posto che a loro compete nel byte ricostituito. Dopo avere ricevuto gli otto bit del dato, occorre verificare il ricevimento del bit di stop. Questo avviene in base alle istruzioni:

```
INC B
IN A, (PORT 0)
RLA
JR NC, LOOPTI
```

Se non c'è carry significa che il bit trasmesso subito dopo i dati era basso, perciò non era un bit di stop. Se ne deduce perciò che qualcosa non ha funzionato come doveva nella comunicazione. In tal caso, avendo inizializzato il registro B ad uno, il controllo viene restituito a LOOPTI che effettua un ulteriore ricerca del bit di stop. Le riprove continuano a ripetersi sintantochè non s'incontra un bit di stop. Quando questa condizione è soddisfatta il controllo è restituito alla routine principale ed il byte ricevuto è caricato nei registri A e C. Si noti che il bit più significativo è resettato. Per convenzione frequentemente nel caso di comunicazione in serie, l'ottavo bit è definito come "indicatore di parità". In molte applicazioni l'ottavo bit è settato quando la parità degli altri sette bit del dato (numero dei bit a uno) è pari e reset-

tato in caso contrario. Questa convenzione è di notevole aiuto nella scoperta degli errori; in quanto, la parte ricevente può verificare la compatibilità tra l'indicatore di parità e quanto è stato ricevuto come bit del dato (7 bit). Tuttavia, come si può osservare, TTYI1 resetta D7 tutte le volte, per cui nell'interfaccia per I/O seriale del Nanocomputer non si effettua alcun controllo della parità.

SUBROUTINE TTYO

Funzione

Pone in uscita su di una linea di trasmissione seriale un byte singolo prelevandolo dal registro C. Il byte viene inviato in uscita con un bit di start e due bit di stop.

Locazione in memoria

TTYO (Per l'indirizzo assoluto si veda l'Appendice A – Tabella A-1).

Dati in ingresso

Il contenuto del registro C costituisce il carattere di otto bit che deve essere trasmesso.

Dati in uscita

Il contenuto del registro C è trasmesso come successione di 11 bit (1 di start, 8 per il dato, 2 di stop) su di un mezzo per comunicazioni seriali.

Registri utilizzati

A e F.

Descrizione

In primo luogo si provvede al salvataggio del contenuto della coppia di registri BC nello stack in modo da poter ripristinare lo stato di partenza prima di restituire il controllo al programma principale. Il flag di carry viene azzerato eseguendo l'istruzione AND A.

Il flag di carry, così "azzerato", viene trasmesso come bit di start. Il registro B è inizializzato col valore decimale 11 per poter contare i bit trasmessi. La trasmissione del byte contenuto nel registro C, è da considerarsi terminata dopo che sono stati trasmessi 11 bit. Il loop LOOPTT, è ripetuto 11 volte. Si osservi come la prima istruzione LOOPTT sia un'istruzione di INGRESSO, davvero inaspettata in una routine di USCITA. La presenza di questa istruzione è giustificata dalla necessità di scoprire quale sia, in quel momento, lo stato logico relativo a ciascuna delle linee associate alla porta 04 che non sono utilizzate per le operazioni di uscita seriale; o precisamente tutte le linee ad accezione di D4. Le istruzioni che vengono successivamente eseguite si limitano a variare lo stato della sola linea D4, mantenendo inalterato lo stato delle altre linee. Le prime tre istruzioni di cui sotto:

```
        RES  4,A
        JR   NC,NXTT
        SET  4,A
NXTT:   OUT  (04H),A
```

settano il bit D4 in modo che esso rispecchi il contenuto del flag di carry. Poichè il flag di carry è inizialmente resettato dall'istruzione AND A, il primo bit trasmesso

è il bit (basso) di start. L'istruzione OUT, in pratica, pone il bit in uscita sulla linea D4. La chiamata della routine BAUD genera un ritardo esattamente pari al periodo di campionamento, in tal modo, la relazione temporale, tra i bit in uscita è esattamente quella attesa dall'unità di ricezione dei dati seriali. Il flag di carry è settato in modo che, l'istruzione successiva RR C, faccia passare per rotazione un bit uno nel bit D7 del registro C. Si tratta cioè di predisporre la trasmissione dei due bit (alti) di stop a chiusura della trasmissione degli otto bit relativi al dato. Si osservi come l'istruzione RR C provochi il passaggio per rotazione nel flag di carry del prossimo bit da trasmettere. Perciò, al momento dell'esecuzione dell'istruzione DJNZ LOOPTT, il flag di carry è definito in modo da predisporre il bit D4 nel modo corretto per la trasmissione. Dopo che sono stati trasmessi nove bit, i bit settati inizialmente dalla istruzione SCF vengono ruotati nel flag di carry e vengono inviati in uscita due bit di stop.

Una volta terminata la trasmissione di tutti gli undici bit, viene ripristinata la coppia di registri BC ed infine restituito il controllo alla routine principale.

Listing completi di commento – TTYI1 e TTYO

```
TTYI1:   IN   A,(04H)    ; Rimani in attesa di un bit basso di start
         RLA             ; Fai ruotare D7 nel flag di carry
         JR   C,TTYI1    ; Se D7 non è basso, rimani in attesa
         CALL BAUDHF     ; Se D7 è basso, controlla che si tratti veramente di un bit
                         ; di start. Aspetta per un tempo pari a metà di un periodo
                         ; di campionamento
         IN   A,(04H)    ; Esamina come si presenta ora D7
         RLA             ; Fai ruotare D7 nel flag di carry
         JR   C,TTYI1    ; Se D7 non è basso torna indietro, al loop di attesa, TTYI1
         LD   B,09H      ; Inizializza B come registro contatore dei bit
LOOPTI:  IN   A,(04H)    ; Fai entrare un bit del dato (La prima volta si tratta ancora
                         ; del bit di start)
         RLA             ; Fai ruotare il bit D7 (il bit in ingresso) nel flag di carry
         RR   C          ; Fai ruotare il flag di carry (il bit ricevuto) nel bit D7 del
                         ; registro C
         CALL BAUD       ; Rimani in attesa esattamente per un tempo pari al periodo
                         ; di campionamento
         DJNZ LOOPTI     ; Decrementa il contatore B di bit. Se è diverso da zero,
                         ; prendi il bit successivo
         INC  B          ; Se B è zero, sono stati ricevuti tutti i bit del dato. Predispo-
                         ; ni B per un'eventuale loop di ritorno a LOOPTI
         IN   A,(04H)    ; Fai entrare quello che dovrebbe essere un bit di stop
         RLA             ; Si tratta di un bit di stop?
         JR   NC,LOOPTI  ; Se il flag di carry è resettato non si tratta di un bit di stop.
                         ; Torna indietro al loop di ingresso dei dati in modo che gli
                         ; ultimi otto bit trasmessi prima del bit di stop siano con-
                         ; tenuti nel registro C
         LD   A,C        ; Se il flag di carry è settato, si tratta di un bit di stop. Trascri-
                         ; vi nell'accumulatore il byte ricevuto in ingresso
         AND  7FH        ; Resetta il bit più significativo – qui non viene utilizzata
                         ; la parità
         LD   C,A        ; Carica il byte in ingresso nel registro C
         RET
```

- -

```
TTYO:    PUSH BC         ; Salva il contenuto della coppia di registri BC
         AND  A          ; Resetta il flag di carry
         LD   B,0BH      ; Carica 11 decimale nel registro B che deve fungere da conta-
                         ; tore dei bit
LOOPTT:  IN   A,(04H)    ; Leggi lo stato dei bit associati alla porta A dei dati del
                         ; PIO N. 1
         RES  4,A        ; Trascrivi il flag di carry nel bit D4
         JR   NC,NXTTT
         SET  4,A
NXTTT:   OUT  (04H),A    ; Poni in uscita il bit D4, senza cambiare lo stato degli altri
                         ; bit nella porta 04.
         CALL BAUD       ; Segna un tempo di ritardo esattamente pari ad un periodo
                         ; di campionamento
         SCF             ; Setta il flag di carry, in modo che i bit alti possano essere
                         ; fatti ruotare in C per poi essere, eventualmente posti in
                         ; uscita come bit di stop
         RR   C          ; Fai ruotare il bit alto al posto del bit più significativo del
                         ; registro C. Fai ruotare, il prossimo bit da trasmettere, nel
                         ; flag di carry
         DJNZ LOOPTT     ; Decrementa B. Se è diverso da zero, torna indietro e invia
                         ; in uscita il bit successivo
         POP  BC         ; Se B è zero, sono stati trasmessi tutti gli 11 bit associati ad
                         ; un byte di otto bit. Ripristina la coppia di registri BC
         RET
```

INTRODUZIONE AGLI ESPERIMENTI

Gli esperimenti che seguono sono stati studiati appositamente per familiarizzarvi con l'unità tastiera/display del Nanocomputer ed il software di sistema utilizzato per gestione di questa.

Esperimento N.	Commenti
1	Illustra il software relativo al display, capace di selezionare i dati da visualizzare, nonchè la posizione
2	Illustra come usare il software di sistema del Nanocomputer per la lettura dei dati inviati da tastiera e la gestione dei display. Le tecniche descritte in questo esperimento troveranno ampia applicazione nei Capitoli 6 e 7

ESPERIMENTO N. 1

Scopo

In questo esperimento viene illustrato praticamente come è possibile gestire i led e i display a sette segmenti appartenenti all'unità tastiera/display del Nanocomputer. Viene descritto un programma di prova del display che visualizza in sequenza ogni possibile configurazione di bit in ciascuna delle posizioni del display. Facendo variare la durata del loop di ritardo tra i cambi di due posizioni è possibile fare sembrare tutti i digit del display pilotati contemporaneamente.

Programma DDRIVE

Codice oggetto	Codice sorgente			Commenti
010500	DDRIVE:	LD	BC,0005H	; B contiene il dato che deve essere visualizza- ; to; C contiene il codice del dispositivo relati- ; vo alla porta di uscita (PIO N. 1 B, dato)
3E00		LD	A,PSEL	; A contiene il selettore della posizione del ; display
00		NOP		; Istruzione non operativa utilizzata in modo ; tale che il programma entrerà nel prossimo ; programma senza dover ricaricare la maggior ; parte dei byte.
ED79		OUT	(C),A	; Invia l'indirizzo del display a HCF4514 cam- ; biando il valore del bit D0
3C		INC	A	
ED79		OUT	(C),A	
3D		DEC	A	
ED79		OUT	(C),A	
ED41		OUT	(C),B	; Uscita del dato
76		HALT		

Passo 1

Caricate il programma (precedente) a partire dalla locazione DDRIVE. Si osservi che la prima istruzione, LD BC,0005H, inizializza il registro C con l'indirizzo della porta B dei dati del PIO N. 1. Il registro B è inizializzato con il dato che deve essere inviato in uscita alla porta 05. La seconda istruzione, LD A,PSEL, carica in accumulatore l'indirizzo del gruppo di sette led o del display a sette segmenti nel quale deve comparire il dato inviato in uscita alla porta 05. Determinati che siano il dato e la posizione di display, l'indirizzo di quattro bit che rappresenta la posizione nel display è introdotto, mediante strobe, nel latch/decodificatore HCF4514B, mentre il byte del dato, contenuto nel registro B, è posto in uscita per essere visualizzato.

Provate ad eseguire questo programma utilizzando come dato il byte 00 e ancora 00 come posizione nel display. Si faccia uso, cioè delle due prime istruzioni:

LD BC,0005H	per predisporre il dato da visualizzare e selezionare la porta di uscita (05H)
LD A,00H	per predisporre la posizione di display.

Che cosa osservate?

Noi abbiamo osservato che tutti i display a sette segmenti sono rimasti spenti, mentre si sono accesi i seguenti led:

BRK, SP, PC, MEM, I/O, ERR e ARS

Questi led rappresentano il gruppo di led facenti capo all'uscita SO del latch/decodificatore HCF4514B. Il risultato è concorde con il fatto che il byte per la posizione di display è 00 poichè l'uscita S0 è attivata tutte le volte che tutti e quattro gli ingressi del dato al latch/decodificatore sono zero. Che significato assume qui il byte del dato?

Il byte del dato assolve a due funzioni. Come prima cosa, la linea D0 abilita oppure disabilita le uscite del latch/decodificatore. Ponendo in uscita 00, alla porta 05 si abilitano le uscite del latch/decodificatore poichè nel byte del dato 00 D0 è basso. I bit da D1 a D7 stabiliscono quali sono i led che si devono accendere. Come abbiamo già avuto occasione di spiegare precedentemente, in seguito all'azione invertente dei transistori del componente Q1, dello schema circuitale della tastiera/display, un bit basso fa accendere un led, mentre un bit alto fa sì che il relativo led rimanga spento. Quindi il byte del dato 00 determina l'accensione di tutti i LED appartenenti alla posizione di display selezionata.

Passo 2

Mantenendo uguale a 00 la posizione di display, cambiate in 01 il byte del dato alla locazione DDRIVE + 2. Cosa pensate che avverrà? Eseguite il programma per controllare la correttezza della vostra previsione. Che cosa osservate?

Noi abbiamo osservato che il display si è spento interamente. Il motivo? La risposta è che il byte del dato, posto in uscita alla porta B dei dati del PIO N. 1, presentava D0 nello stato logico 1. Dal momento che PB0 è collegato all'ingresso inhibit di HCF4514B, attivo nello stato ALTO, essendo alto, ha inibito le uscite del latch/decodificatore il bit meno significativo del byte del dato. Ne consegue una mancata attivazione di qualsiasi posizione di display, ossia uno spegnimento totale.

Programma DISTST

```
Codice
oggetto   Codice sorgente                Commenti
010500    DISTST: LD   BC,0005H          ; B contiene il dato da visualizzare, C contie-
                                         ; ne il codice corrispondente al dispositivo di
                                         ; uscita
AF        DATLP:  XOR  A                 ; A contiene la posizione di visualizzazione
160A              LD   D,0AH             ; D è il contatore della posizione del display
ED79      OUTPUT: OUT  (C),A             ; Invia l'indirizzo del display ad HCF4514B
                                         ; facendo variare il bit D0
3C                INC  A
ED79              OUT  (C),A
3D                DEC  A
ED79              OUT  (C),A
ED41              OUT  (C),B             ; Uscita del dato
3C                INC  A                 ; Incrementa la posizione del puntatore per
                                         ; puntare alla prossima posizione display
3C                INC  A
CDE301            CALL DELAY             ; Introduce una pausa in modo da lasciare fis-
                                         ; so il display per un breve intervallo di tempo
15                DEC  D                 ; Decrementa la posizione del contatore
20EE              JR   NZ,OUTPUT         ; Se D non è zero ritorna indietro per inviare
                                         ; in uscita un byte sulla prossima posizione
                                         ; del display
04                INC  B                 ; Se sono state controllate tutte le posizioni
                                         ; del display, aggiorna il dato in uscita
04                INC  B                 ; Inizia di nuovo con un altro
18E7              JR   DATALP            ; byte di dato
D5        DELAY:  PUSH DE                ; Salva DE
16F0              LD   D,0F0H            ; Byte di temporizzazione
```

```
CDF2F9     DREGL:  CALL BAUD        ; BAUD è una routine del sistema operativo
                                    ; che introduce un ritardo pari al periodo di
                                    ; campionamento. La lunghezza del periodo è
                                    ; fissata tramite un byte memorizzato in me-
                                    ; moria.
                                    ; Il tempo di ritardo della subroutine DELAY
                                    ; sarà uguale a 16 (decimale) periodi di cam-
                                    ; pionamento
15                 DEC  D
20FA               JR   NZ,DREGL
D1                 POP  DE          ; Ristabilisce il contenuto di DE
C9                 RET
```

Passo 3

La tabella che segue elenca un certo numero di valori interessanti per i dati e le posizioni, con una descrizione delle relative visualizzazioni da noi osservate. Verificate che la tabella rispecchi, esattamente, anche le vostre osservazioni.

Dato (DDRIVE + 2)	Posizione (DDRIVE + 4) = PSEL	Descrizione della visualizzazione risultante
00	00 e 20	; BRK, SP, PC, MEM, I/O, ERR, ARS tutti accesi; spenti tutti i display a 7 segmenti. I byte di posizione 00 e 20 generano la medesima visualizzazione in quanto i bit da D1 a D4 sono gli unici ad essere posti in ingresso al latch/decodificatore HCF4514B. Per questo motivo, 00, 20, 40, 60, 80, A0, C0 ed E0 selezionano tutti la stessa posizione di display.
01	un byte qualsiasi	; Tutti i display spenti
00	01	; BRK, SP, PC, MEM, I/O, ERR, ARS tutti accesi; spenti tutti i display a 7 segmenti. Osservazione: avviene la stessa visualizzazione come quando il byte del dato e quello della posizione sono uguali a 00 poichè la posizione di display è determinata dai bit che vanno da D1 a D4
00	02	; IY, IR, AF, BC, DE, HL ed IX sono tutti accesi; spenti tutti i display a 7 segmenti.

Dato	Posizione	7	6	5	4	3	2	1	0
00	04					8			
00	06						8		
00	08							8	
00	0A								8
00	0C	8							
00	0E		8						
00	10			8					
00	12				8				
00 00	14, 16, 18 1A, 1C, 1E	Spenti tutti i display							
80	04					U			
02	04					0			

(Il bit D7 corrisponde al segmento a)
(Il bit D1 corrisponde al segmento g)

Passo 4

Caricate il programma (precedente) a partire dalla locazione DISTST. Questo programma può essere utilizzato per la prova dei display del Nanocomputer. In ciascuna delle dieci posizioni del display appare visualizzata una delle possibili combinazioni di sette bit. Poichè i possibili caratteri visualizzati sono 2^7, proponendovi di provare completamente la vostra unità di visualizzazione, dovreste guardare e controllare 10 caratteri visualizzati per ben 128 volte. Poichè ciò comporterebbe un impiego di tempo non indifferente, per non parlare della noia che fatalmente vi assalirebbe, questo programma è da intendersi soprattutto come un'esercitazione sulla gestione del display del Nanocomputer piuttosto che come un serio collaudo di esso.
Si esegua il programma. Che cosa osservate?

Noi abbiamo osservato varie sequenze dei dieci display:

1° sequenza: Accensione simultanea dei LED BRK, SP, PC, MEM, I/O, ERR, ARS.
Accensione simultanea dei LED IY, IR, AF, BC, DE, HL, IX
Contando da sinistra da 1 ad 8; i display a sette segmenti si sono accesi visualizzando un otto, nel seguente ordine: 5, 6, 7, 8, 1, 2, 3, 4

2° sequenza: Accensione simultanea dei LED BRK, SP, PC, MEM, I/O, ERR
Accensione simultanea dei LED IR, AF, BC, DE, HL, IX
Contando da sinistra da 1 ad 8, i display a sette segmenti si sono accesi visualizzando uno zero nel seguente ordine: 5, 6, 7, 8, 1, 2, 3, 4.

ecc.

Le uscite visualizzate del display da voi osservato dovrebbero essere le stesse.

Riguardo a quest'ultimo programma, va notato che sia il byte del dato che quello della posizione sono incrementati due volte ad ogni ciclo. Dal momento che il bit D0 non influisce in alcun modo né sul display né sulla posizione selezionata si rende necessario un doppio incremento. Per quanto concerne il byte del dato, il doppio incremento evita di porre in uscita un bit D0 alto verso il latch/decodificatore HCF4514B. Questo è importante in quanto al variare dei dati i display sarebbero altrimenti disabilitati a cicli alternati.

Passo 5

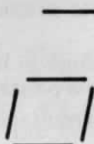
Come abbiamo già avuto occasione di osservare nel corso di questo capitolo, qualsiasi visualizzazione, consistente in digit multipli a sette segmenti e/o in gruppi di led, è ottenuta scandendo in rapida sequenza tutti e dieci i display. Si può osservare l'effetto prodotto sul display dai repentini cambiamenti della posizione variando la durata del ritardo introdotto dalla subroutine DELAY. A questo scopo, si sostituisca F0 con 01 nella subroutine DELAY il byte di temporizzazione. Si cambino inoltre in NOP (esadecimale: 00) le due istruzioni INC B, in modo da fissare un unico valore per i dati. Eseguite ora il programma partendo dalla locazione DISTST. Che cosa osservate?

Noi abbiamo osservato che tutti i led si sono accesi e che tutte le posizioni dei display a sette segmenti presentavano il numero 8. Non abbiamo osservato nessuna intermittenza, proprio come se tutti i display fossero pilotati simultaneamente.

Passo 6

Si sostituisca con il valore 54 il byte del dato nell'istruzione LD BC,0005H, cambiando cioè l'istruzione in LD BC,5405H. Eseguite di nuovo il programma. Che cosa avete osservato questa volta?

Noi abbiamo osservato che si sono accesi i seguenti led: BRK, MEM, SP, ARS, IR, BC, HL ed IY. Su tutti i display a sette segmenti è comparsa la figura:



ESPERIMENTO N. 2

Scopo

Lo scopo di questo esperimento è quello di illustrarvi come usare il software di sistema, presente nel sistema operativo del Nanocomputer, per poter accettare dati provenienti dalla tastiera o generare delle visualizzazioni predeterminate sull'unità tastiera/display.

Programma KBTST

Codice oggetto		Codice sorgente		Commenti
CD9DF9	KBTST:	CALL	CHECKB	; Controlla se sono stati premuti i tasti
28FB		JR	Z,KBTST	; Se flag Z = 1 equivale a nessuna digitazione
CDDBF8	GETNO:	CALL	KBSCAN	; Se Z = 0 sono stati premuti uno o più tasti. ; Vedi se è uno solo e quale è.
38F6		JR	C,KBTST	; Se il flag C = 1 significa che è stato premuto più di un tasto
32E20F		LD	(DATAL),A	; Se il flag C = 0 significa che è stato premuto un solo tasto il cui numero identificativo è contenuto nel registro A. Visualizza il numero esadecimale corrispondente al tasto nella zona dati del display
08		EX	AF,AF'	; Predispone per la chiamata di CONVDI
3EFC		LD	A,0FCH	; Visualizza solamente le cifre dei dati
08		EX	AF,AF'	
11E50F		LD	DE,ADDH	
21B90F		LD	HL,ADD7-1	
CD7CFA		CALL	CONVDI	; Converte il numero del tasto per la visualizzazione
CD09F9	DISPLAY:	CALL	DISPL	; Visualizza il numero del tasto
CD09F9		CALL	CHECKB	; Controlla se ci sono tasti premuti

```
28F8    JR   Z,DSPLAY   ; Se non ci sono tasti premuti continua la
                        ; visualizzazione
18E1    JR   GETNO      ; Se viene premuto un tasto prende il numero
                        ; corrispondente
```

Passo 1

Si carichi il programma (precedente) a partire dalla locazione KBTST. Tale programma utilizza una buona parte delle routine principali del sistema operativo del Nanocomputer adibite alla gestione della tastiera/display:

CHECKB	per il rilevamento di un tasto premuto
KBSCAN	per accertarsi che è stato premuto un solo tasto e identificatare tale tasto
CONVDI	per effettuare la traduzione di dati memorizzati in binario o esadecimali in un codice idoneo a pilotare i display a sette segmenti
DISPL	per la visualizzazione "simultanea" dei dati sui led nonchè sui display a sette segmenti

Il flusso logico del programma è il seguente:

1. Controlla che sia stato premuto un tasto.
2. Controlla che sia stato premuto un unico tasto e, in caso affermativo, lo identifica.
3. Visualizza, nei due digit più a destra del display a sette segmenti il numero corrispondente al tasto premuto.
4. Continua a visualizzare il numero, relativo al tasto appena premuto, fino a quando viene premuto un altro tasto. Quando si riscontra quest'ultima condizione visualizza il numero corrispondente.

Fate eseguire il programma, premendo, lievemente e rapidamente, il tasto GO. Che cosa osservate?

Noi abbiamo osservato che tutti i display si sono spenti. Se avessimo avuto la mano un po' più "pesante" nel premere il tasto GO, in corrispondenza dei due digit di destra del display avremmo osservato un 19.

Passo 2

Premete i tasti 0, 1, 2, 3, . . ., F. Che cosa osservate?

Sulla unità tastiera/display, noi abbiamo osservato le seguenti visualizzazioni 00, 01, . . ., 0F. Inoltre, il display si spegneva regolarmente dopo che avevamo premuto un tasto. Premendo nuovamente lo stesso tasto compariva immediatamente la visualizzazione del digit esadecimale ad esso associato. La nostra spiegazione, per questo comportamento un po' "bizzarro" è la seguente: il programma esegue sempre la lettura del tasto premuto in due tempi, una volta nella routine CHECKB e la seconda in KBSCAN. Se il rilascio del tasto da parte vostra avviene nell'intervallo compreso tra queste due letture successive, il programma non riconosce come valido tale ingres-

so. Per cui il controllo torna al loop KBTST, durante l'esecuzione del quale tutti i display rimangono spenti, mentre la routine attende la digitazione di un tasto. Nella Tavola 5-3 è riportata una tabella completa dei tasti e dei numeri visualizzati ad esso corrispondenti.

Passo 3

Prima della chiamta di CONVD! si osservi come sono inizializzati il registro alternativo A' e le coppie di registri DE ed HL. Cambiando il contenuto del registro A', osserverete delle differenze nei digit a sette segmenti di volta in volta visualizzati. Cambiate, per esempio, A' in FA ed eseguite il programma. Che cosa osservate?

Noi abbiamo osservato che il primo digit di destra corrispondeva ancora al digit meno significativo del tasto premuto. Il digit adiacente alla sua sinistra era spento, mentre il secondo digit verso sinistra era sempre 1. Potreste avere osservato, in corrispondenza del terzo digit a partire dal fondo la visualizzazione di un digit diverso in quanto il suo valore dipende da ciò che si trova contenuto in DATAH.

Tavola 5-3. Tasti e numeri corrispondenti della tastiera del Nanocomputer.

Tasto	Numero visualizzato	Tasto	Numero visualizzato
0	00	→	10
1	01	←	11
2	02	ST	12
3	03	LA	13
4	04	2ND	14
5	05	SS	15
6	06	INC	16
7	07	LD	17
8	08	ARS	18
9	09	GO	19
A	0A	BRK	1A
B	0B	DP	1B
C	0C		1C
D	0D	BREAK	Nessuna variazione rispetto alla visualizzazione precedente.
E	0E		
F	0F	RESET	Uscita dal programma per entrare nel sistema operativo.

NOTA FINALE PER LO SPERIMENTATORE

Un po' per divertimento, vi proponiamo di provare il programma, corrispondente a quello utilizzato per visualizzare la frase

SGS-ATES NANOROUTINES RELEASE LOADED CIAO

che appare allorchè la ROM usata negli esperimenti è caricata nell'area di RAM della scheda. Lo stesso programma può essere fatto girare dal punto di partenza NANOR2 in RAM della locazione F6EC in ROM.

I dati in corrispondenza di STRING possono essere da voi cambiati in modo da scrivere qualunque frase di vostra scelta: la stringa può essere di lunghezza qualsiasi, a patto che termini con il byte 01H, 9 byte prima della fine.

CAPITOLO 6

TECNICHE DI INTERRUZIONE

INTRODUZIONE

Si immagini la seguente sequenza di eventi:

1. State leggendo un libro.
2. Squilla il telefono.
3. Segnate con un segnalibro, od in altro modo, il punto a cui siete arrivato e sollevate il ricevitore del telefono.
4. Pronunciate la parola "PRONTO" per segnalare che siete a disposizione per conversare con la persona che vi ha chiamato.
5. Ha inizio la conversazione.
6. Suonano alla porta di ingresso.
7. Avvertite la persona al telefono di attendere un momento.
8. Andate ad aprire la porta.
9. Avete un colloquio con la persona che si è presentata alla porta, e pervenite ad una qualche conclusione con costui o costei.
10. Tornate al telefono, riprendete la conversazione con l'altra persona all'apparecchio, portando a termine la discussione.
11. Tornate alla lettura del vostro libro, rintracciando il punto in cui siete stati interrotti.

La parte da voi impersonata in questa scena è quella di una risorsa da ripartire tra tre compiti: leggere il libro, parlare al telefono e intrattenersi con la persona che si è presentata alla porta. In ogni singolo momento, il compito del quale vi stavate occupando era uno, ed uno solo, dei tre da svolgere. I vari passaggi dall'uno all'altro compito sono innescati da due meccanismi differenti:

a. Interruzioni, nel nostro caso lo squillo del telefono oppure il campanello che suona.
b. Completamento dei vari compiti, ad esempio, terminare una conversazione.

Questi passaggi, o transizioni, sono nidificati, nel senso che, al verificarsi di una interruzione, il compito di cui ci si sta occupando è lasciato in sospeso e tutte le risorse sono rivolte al compito che determina l'interruzione sino a che quest'ultimo non sia stato portato a termine, per riprendere, a questo punto, il compito precedentemente interrotto. Le transizioni in seguito ad interruzione sono tutte caratterizzate da questa successione di eventi:

a. Ha luogo l'interruzione
b. Cessa ogni attività in corso e si procede alla memorizzazione dello stato attuale per poterlo ripristinare più tardi.
c. L'interruzione è riconosciuta inviando una qualche segnalazione al richiedente della medesima.
d. Ha inizio il servizio dell'interruzione, che prosegue sino a che non intervenga un'altra interruzione oppure che il servizio stesso non sia portato a termine.

Le transizioni in seguito a completamento di un compito sono tutte caratterizzate da questa successione di eventi:

a. E' terminato il servizio dell'interruzione.
b. E' richiamato il punto di arresto dell'operazione precedentemente interrotta.
c. Le risorse sono di nuovo rivolte all'espletamento del compito che era stato interrotto.

La scena che abbiamo immaginato potrebbe anche svolgersi diversamente. Si supponga infatti, che per un motivo qualsiasi, voi desideriate non essere interrotti durante la lettura del vostro libro. Piuttosto che prendere atto dell'interruzione rappresentata dal telefono o dal campanello dell'ingresso, potete, ad esempio, fingere di non sentirli, avendo deciso di non occuparvene. Sarebbe questo il caso di "interruzioni disabilitate". Naturalmente, vi è almeno una interruzione che, con ogni probabilità, non potete ignorare tanto facilmente, come, ad esempio, un fusibile bruciato che vi lascerebbe completamente al buio. Una interruzione di questo tipo corrisponderebbe ad una *"interruzione-non mascherabile"* in quanto non sarebbe possibile, "mascherarla", ignorandone le conseguenze.

Per poter proseguire è, infatti, necessario prestare la opportuna assistenza (sostituire il fusibile). Alle interruzioni non mascherabili compete un grado di priorità più elevato che non a quelle che possono essere ignorate *(interruzioni mascherabili)*.
E' possibile associate ad ogni interruzione mascherabile una priorità, in modo da consentire la prosecuzione delle operazioni a priorità più elevata senza possibilità di interruzione da parte di quelle a priorità inferiore. E' chiaro, infatti, che, quando si è impegnati in una conversazione telefonica internazionale, è meglio non subire interruzioni da parte di visitatori occasionali.

Se, invece, acconsentiste a lasciarvi interrompere da ogni piccola distrazione, perdereste tutto il vostro tempo a gestire le interruzioni, senza riuscire a leggere una sola riga. Sareste, in tal caso, *interrupt bound* (letteralmente: immobilizzato a causa delle interruzioni; passereste, cioè, di continuo da un'interruzione all'altra, senza potere fare niente altro). Se fosse consentito sovrapporre le interruzioni, l'una sopra all'altra, in modo tale da non riuscire mai a giungere al completamento di nessuna operazione, si potrebbe parlare di *thrashing* (condizione di attività dispersive dovuta all'incessante stillicidio di sempre nuove incombenze).
Il nodo centrale della discussione precedente si configura, dunque, come problema di *ripartizione delle risorse (resource sharing)*. Voi rappresentate, infatti, la risorsa che deve essere concentrata su differenti compiti tra di loro in concorrenza, dato che ciascuno di essi esige il vostro interessamento per essere svolto. Poichè le risorse a disposizione sono limitate, c'è *conflittualità* tra i vari compiti; è desiderabile che tali *conflitti* vengano risolti in modo tale da massimizzare l'efficienza globale del sistema.

Questa stessa analisi può essere ritenuta valida se riferita al problema di come suddividere le risorse della CPU tra più operazioni tra di loro in concorrenza. Nel caso di un grande sistema di elaborazione, a queste operazioni tra di loro in concorrenza possono corrispondere utenti diversi che, attraverso terminali remoti, chiedono l'esecuzione di propri lavori. Nel caso di un microcomputer, tali compiti possono consistere in operazioni di trasferimento di dati tra diversi dispositivi di I/O, secondo una tecnica detta *interrupt driven I/O* (I/O guidati da interruzioni). Nella prima parte di questo capitolo tratteremo alcune tecniche alternative utilizzate per la ripartizione delle risorse della CPU. Alle tecniche di gestione delle interruzioni spetta giustamente un posto di particolare rilievo in quanto esse sono tra le più utilizzate nelle applicazioni con microcomputer. Perciò, alternando cenni di teoria ad esperimenti pratici, illustreremo le particolarità delle diverse tecniche di interruzione con la CPU Z80.

OBIETTIVI

Al termine di questo capitolo sarete in grado di:

- Conoscere il significato dei termini *interrupt* (interruzione), *polling* (interrogazione ciclica) e DMA *(direct memory access:* accesso diretto in memoria), che indicano tecniche utilizzate per la ripartizione delle risorse della CPU.
- Confrontare i pregi rispettivi dell'elaborazione a polling e di quella a interruzioni, con riferimento al grado di utilizzazione delle risorse della CPU ed all'efficienza.
- Conoscere il significato dei termini di interruzioni *mascherabili* e *non-mascherabili.*
- Sapere a cosa corrispondono questi tre abituali metodi di elaborazione mediante interruzioni: *interruzioni su di una singola linea, interruzioni a più livelli* ed *interruzioni vettorizzate.*
- Conoscere il significato dei termini *switching del contesto, thrashing* e *codice rientrante.*
- Interpretare e montare i circuiti necessari per implementare i tre modi di interruzione della CPU Z80.

CARATTERISTICHE FUNZIONALI DEI MICROCOMPUTER

In questo manuale ed in quello precedente dedicato al software, il modo di operare dello Z80 è stato descritto nell'ipotesi ricorrente che l'esecuzione di un programma, una volta avviata, prosegua sino al suo completamento. Questo modo di operare del computer, in base alla quale tutte le sue risorse sono impiegate durante l'esecuzione di una operazione, senza essere distolte sino a che quest'ultima non sia terminata, è, tra tutti i possibili, quello più semplice. Per l'esattezza, è bene osservare che già avete avuto occasione di incontrare un esempio di gestione di interruzioni, benchè la relativa trattazione mancasse di riferimenti espliciti al proposito. Il tasto di esecuzione passo-passo (single step) sfrutta infatti le caratteristiche delle interruzioni dello Z80, allo scopo di ottenere il continui andirivieni del controllo tra il programma utente ed il sistema operativo del Nanocomputer. Poichè le tecniche utilizzate in questo caso dipendono dalle proprietà del chip PIO, tralasceremo, per ora, di occuparcene ulteriormente.

I modi di operare fondamentali dello Z80, dei quali ci si occuperà in questo capitolo, sono due:

SOFTWARE CONTROLLED PROCESSING *(Elaborazione controllata da software)*

Modalità di elaborazione secondo la quale qualunque transizione da un compito ad un altro avviene interamente sotto il controllo del software. Esempi di transizione controllate da software sono rappresentati dalle chiamate di subroutine, i ritorni da esse ed i salti. Una tecnica di cui ci si vale abitualmente per implementare transizioni controllate da software corrisponde al polling.

HARDWARE AND SOFTWARE CONTROLLED PROCESSING *(Elaborazione controllata da hardware e software)*

Modalità di elaborazione secondo la quale le varie transizioni da un compito all'altro avvengono sotto il controllo o dell'hardware oppure del software. Esempi di transizioni controllate da hardware sono rappresentati dalle interruzioni e dai trasferimenti di dati in DMA. I chip periferici Z80, denominati circuiti integrati PIO e DMA, sono appositamente progettati per applicazioni che prevedono, rispettivamente, interruzioni e DMA.

E' opportuno, a questo punto, chiarire alcuni dei termini di cui abbiamo fatto uso:

Polling (Interrogazione ciclica)

Con polling si intende una tecnica software utilizzata per stabilire se un dispositivo esterno necessita di assistenza. Tale dispositivo risulta collegato alla CPU tramite una linea di comunicazione che può essere esaminata ciclicamente per accertare se c'è una richiesta di servizio.

Interrupt (Interruzione)

Un'interruzione consiste in un segnale inviato in un particolare della CPU per indicare che un dispositivo esterno necessita di servizio, ossia, delle risorse della CPU. Di norma, nel corso dell'esecuzione di ciascuna istruzione, la CPU esegue il test del proprio ingresso di interruzione per controllare se si trova nello stato attivo. In caso affermativo, l'istruzione in corso è portata a termine prima di servire l'interruzione.

Direct memory access (DMA) (Accesso diretto in memoria)

L'impiego di questa tecnica è frequente nel caso si desiderino trasferire dei blocchi estesi di dati tra la CPU ed un dispositivo periferico. Previa autorizzazione della CPU, il trasferimento si compie sotto la direzione esclusiva di un circuito esterno destinato a questo scopo. Ne consegue che nell'operazione di trasferimento non risulta praticamente impegnata nessuna delle risorse della CPU. Per completezza, occorre tuttavia prevedere una certa diminuzione delle prestazioni generati, in conseguenza del fatto che la CPU deve spartirsi con il dispositivo DMA l'uso dei bus degli indirizzi dei dati e di controllo interessati dal trasferimento dei dati.

Un esempio tipico, che illustra egregiamente la differenza tra polling, tecniche di interruzione ed accesso diretto, può considerarsi il seguente:

Nel caso che abbiate l'abitudine di ricevere le telefonate con la tecnica di *polling*, non dovreste far altro che staccare ogni tanto il ricevitore per chiedere "C'è per caso qualcuno che vuol parlare?"

Con il sistema di comunicazioni telefoniche basate sulla *tecnica di interruzione* attendereste di udire squillare il telefono per sollevare, quindi il ricevitore.

Se, per ultimo, aveste deciso di ricevere telefonate servendovi della tecnica di *accesso diretto in memoria,* registrereste con un registratore i vari messaggi telefonici, in modo da poterli ascoltare al momento opportuno. Vi è da osservare come questo sistema non sia particolarmente indicato per gestire frequenti messaggi brevi tra voi ed il vostro interlocutore. La tecnica DMA non è infatti quella ideale allorchè si vogliano soddisfare delle necessità di questo tipo.

Le tecniche qui descritte corrispondono a tre diversi modi di affrontare il problema della distribuzione delle risorse della CPU tra un certo numero di utenti potenziali. La tecnica di polling coincide con il criterio del massimo accentramento, in quanto è la CPU ad assumere l'iniziativa di interpellare singolarmente ciascuno degli utenti al fine di appurare l'eventuale necessità di un servizio. Nel caso ci si imbatta in un utente che richiede un servizio, prima di interpellare il prossimo utente, è assicurato il completo soddisfacimento delle necessità del primo. Appare dunque chiaro che i conflitti tra i diversi utenti sono risolti interpellando quest'ultimi uno per volta. L'elaborazione mediante interruzioni esige, al contrario, che l'utente esplici una funzione più attiva in fase di richiesta delle risorse. Il meccanismo che procura il servizio è, intatti, quello dell'interruzione. In un sistema dove non è associata priorità alle interruzioni, la CPU convalida qualunque interruzione al momento stesso del suo manifestarsi e dà subito inizio al relativo servizio. L'interruzione più recente

(l'ultima in ordine di tempo) continua ad essere servita sino al completamento delle operazioni necessarie oppure sino a che non si verifichi un'altra interruzione. Tuttavia, per risolvere, tra l'altro, il problema che nascerebbe nel caso che due utenti inoltrassero richiesta di servizio nello stesso preciso istante, quasi tutti i sistemi esistenti associano ad ogni potenziale sorgente di interruzione una priorità. L'utente al quale compete il grado di priorità più elevato ha così assicurato il servizio sino al suo completamento oppure sino al verificarsi di un'interruzione richiesta da un utente a priorità ancora più elevata. Questo modo di affrontare il problema della "allocazione delle risorse" è meno centralizzato del polling, in quanto sono gli utenti ad assumersi l'iniziativa di stabilire la comunicazione con la CPU. Il metodo DMA affronta il medesimo problema, recando in aggiunta un potente insieme di nuove risorse, corrispondenti al controller DMA. A differenza delle tecniche di polling e d'interruzione, il DMA non si propone come una tecnica di uso generale, poichè la sua unica applicazione degna di rilievo è quella di eseguire trasferimenti di estesi blocchi di dati tra la memoria della CPU ed altre unità periferiche. Pertanto il DMA serve, più che altro, a potenziare le risorse complessive fornite dal sistema con nuove capacità molto specializzate e dirette solo ad una parte degli utenti.

Dall'esempio precedente a proposito del telefono potete desumere immediatamente più di un vantaggio inerente alla tecnica di interruzione rispetto al polling. Il principale svantaggio del polling è di impegnare parte delle risorse anche se non vi è richiesta di servizio. Un altro difetto consiste nel fatto che il tempo di risposta alla richiesta di servizio può assumere una durata eccessiva. Nell'esempio relativo al telefono può, infatti, succedere che, se il campionamento della vostra linea telefonica avviene soltanto una volta ogni ora, un eventuale interlocutore può attendere ben 59 minuti prima di poter parlare con voi.

Rispetto al polling le procedure basate sull'interruzione presentano alcuni notevoli vantaggi. Il primo di essi consiste nel fatto che non vi è spreco di risorse per verificare se c'è una richiesta di servizio, in quanto la ripartizione delle risorse è fondata rigorosamente sulle necessità. Il servizio, inoltre, può essere prestato non appena se ne manifesta la necessità, cosicchè il tempo di risposta si riduce al minimo. Di contro a questi pregi, le procedure basate sull'interruzione hanno il grande svantaggio del software estremamente complesso, poichè dal momento che le varie interruzioni possono interrompere un programma in un punto qualunque è necessario del software in più che garantisca la ripresa dell'esecuzione del programma una volta prestato il servizio richiesto. Il fatto che le interruzioni accadono casualmente complica notevolmente la simulazione, l'individuazione del software. Un ulteriore difetto dipende dalla frequente presenza di strutture hardware esterne, di costo non trascurabile, per determinare la priorità e identificare i dispositivi che richiedono l'interruzione. Negli esperimenti che sarete invitati ad effettuare sono contemplate parecchie delle situazioni esemplari precedentemente considerate.

La tecnica di accesso diretto in memoria, che interessa il trasferimento di grandi blocchi di dati tra la CPU e le varie periferiche, comporta l'aggiunta di una risorsa notevole, in opposizione al criterio di suddividere le risorse della CPU. Anche se l'uso dei bus dei dati, degli indirizzi e di controllo è in comune tra la CPU ed il controller DMA, questa tecnica equivale grosso modo all'aggiunta di un'altra CPU dedicata esclusivamente alla gestione dei trasferimenti di I/O. Se, da un lato, il DMA ha il grande merito di non utilizzare per niente le risorse della CPU, spesso, tuttavia, considerazioni, legate al corso ed alla complessità della sua implementazione, ne sconsigliano l'adozione.

TIPI FONDAMENTALI DI INTERRUZIONE

Tutte le interruzioni possono essere distinte in *mascherabili* oppure *non-mascherabili*. Chiariamo innanzitutto questi termini:

Interruzione mascherabile

Un'interruzione si dice mascherabile se è possibile programmare da software la CPU, affinchè questa ignori la interruzione. La CPU Z80 è dotata di una linea di interruzione mascherabile, contrassegnata con il nome INT. La istruzione da impartire alla CPU Z80 perchè essa non riconosca alcuna interruzione mascherabile è DI, sigla di Disable Interrupt (disabilita le interruzioni). L'istruzione che si utilizza per abilitare il riconoscimento delle interruzioni mascherabili è EI, sigla di Enable Interrupt (abilita le interruzioni). Di entrambe queste istruzioni si daranno in varie occasioni esempi e chiarimenti.

Interruzione non-mascherabile

Una interruzione è detta non-mascherabile quando la CPU non può ignorarla in qualunque circostanza. La linea, unica, di interruzione non-mascherabile della CPU è chiamata NMI.

Per la maggior parte dei dispositivi che possono ottenere il servizio della CPU su interruzione, l'uso di quelle non-mascherabili consente di avere la risposta più veloce possibile. La prima operazione della routine di gestione di una interruzione consiste nel salvataggio dello stato della CPU in modo che l'esecuzione del programma interrotto possa riprendere correttamente una volta completata l'operazione che ha determinato l'interruzione. Mentre sono in corso di esecuzione le varie istruzioni necessarie per salvare lo stato della CPU, è buona norma disabilitare le interruzioni mascherabili.

L'applicazione principale delle interruzioni non-mascherabili riguarda il riconoscimento della caduta dell'alimentazione. Il microprocessore Z80 è alimentato da una tensione di +5 Volt DC derivata da una linea di rete a 220/110 Volt AC. Un circuito rilevatore della caduta della rete, concepito in modo da innescare un'interruzione non-mascherabile allorchè la linea AC scende al di sotto dell'80%del valore nominale, è in grado di preavvertire la CPU con un buon anticipo, prima che essa non funzioni più correttamente per effetto della caduta dell'alimentatore. Poichè, prima che la tensione di alimentazione scenda sotto a +5 Volt, dovranno trascorrere alcuni millisecondi, la CPU ha tutto il tempo di salvare il proprio stato in modo da riprendere correttamente al ritorno dell'alimentazione. L'estrema importanza di questo accorgimento è evidente se si pensa ad un'applicazione di controllo di processo in cui una valvola lasciata aperta (stato ON) può significare l'allagamento dell'edificio o, peggio ancora, un'esplosione nella fabbrica.

Internamente alla classificazione primaria delle interruzioni in mascherabili e non-mascherabili è possibile definire diverse categorie in base al meccanismo di gestione delle interruzioni, ossia al modo in cui la CPU ed il dispositivo che causa l'interruzione si scambiano alcune indispensabili informazioni come:

Dal dispositivo alla CPU——

a. Richiesta di servizio
b. Identificazione, in modo che la CPU sia edotta su quale sia il servizio da effettuare.

Dalla CPU al dispositivo——

a. Riconoscimento dell'interruzione
b. Dati diversi, a seconda delle necessità

I tipi possibili di interruzione sono tre: *single-line* (su singola linea), *a più livelli* e *vettorizzata*. A titolo di chiarimento, ne riportiamo la definizione:

Interruzione single-line

Esiste un particolare ingresso della CPU al quale vengono presentate le richieste di interruzione. Nel caso che il sistema preveda un unico dispositivo abilitato alla inter-

ruzione, questo è il modo più semplice e diretto di implementare la gestione delle interruzioni. Se i dispositivi sono più d'uno, essi devono inserirsi su questa linea in modo tale che un'interruzione dovuta ad uno qualunque di essi sia comunicata direttamente alla CPU. Una volta riconosciuta l'interruzione la CPU determina il dispositivo che ha generato l'interruzione, interrogando i dispositivi presenti.

Interruzione a più livelli

Sulla CPU sono presenti più di un ingresso per ricevere le richieste di interruzione. Ciascuno dei dispositivi autorizzati all'interruzione è, in questo caso, collegato ad uno di questi ingressi. Ne consegue che la CPU identifica rapidamente il dispositivo che ha generato l'interruzione.

Interruzione vettorizzata

L'interruzione non comporta la sola richiesta di interruzione, ma prevede altresì un identificatore che consente alla CPU di passare direttamente all'apposita routine di gestione dell'interruzione.

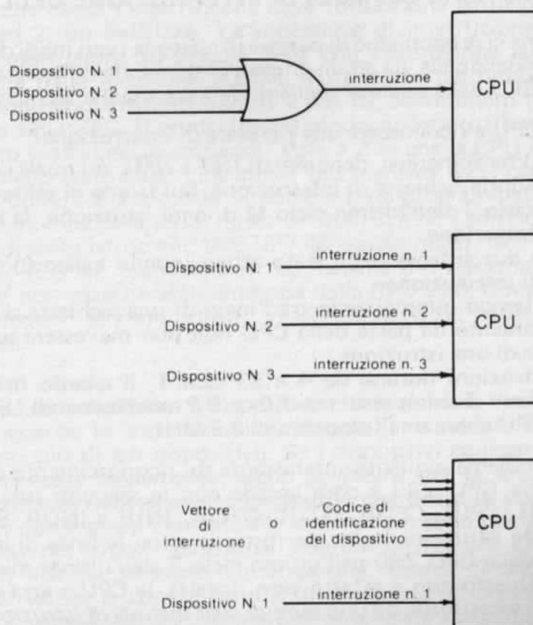


Figura 6-1. I tre tipi di interruzione.

I tre tipi di interruzione appena descritti sono rappresentati schematicamente in Figura 6-1.

La semplicità ed il costo modesto per la sua implementazione sono la ragione della forte diffusione del tipo a linea singola nelle applicazioni con microprocessori. Il numero di dispositivi che si possono inserire su di una singola linea di interruzione è praticamente illimitato. L'esempio di Figura 6-1 mostra tre dispositivi. E' comunque importante osservare che, con l'incremento del numero dei dispositi, aumenta

di riflesso, il tempo necessario a interrogarli tutti e, di conseguenza, il tempo di risposta all'interruzione.

L'interruzione a più livelli costituisce una buona tecnica di interruzione a patto che il chip del microprocessore disponga di un numero sufficiente di pin di interruzione, condizione che si verifica raramente, in quanto i chip degli attuali microprocessori di norma sono dotati, al massimo, di quattro pin adibiti alle interruzioni.

La tecnica di interruzione vettorizzata consente la selezione immediata della routine di gestione dell'interruzione relativa al dispositivo che l'ha generata. Nel caso dei microprocessori ad otto bit, l'impulso di interruzione inviato alla CPU è seguito da un codice ad otto bit avente la funzione di identificare il dispositivo. Nella pratica, il codice ad otto bit corrisponde o ad un indirizzo, o al puntatore ad un indirizzo oppure ancora al primo byte di un'istruzione.

La CPU Z80 ammette sia interruzioni single-line che vettorizzate. In questo ultimo modo di interruzione è previsto un codice di otto bit che identifica il dispositivo, e che consiste in un puntatore all'indirizzo di inizio della routine di servizio dell'interruzione oppure nel primo byte di una istruzione dello Z80.

CARATTERISTICHE DEI MODI DI INTERRUZIONE DELLA CPU Z80

In questa sezione ci proponiamo di passare in rassegna i vari modi di interruzione della CPU Z80, rispondendo via via ad un gruppo di domande d'interesse generale, valide per una qualsiasi CPU della quale si vogliano definire modi e tecniche di interruzione.

1. Come fa la CPU a riconoscere una richiesta di interruzione?

   La CPU Z80 ha 2 ingressi, denominati $\overline{INT}$ e $\overline{NMI}$, sui quali i dispositivi esterni inviano le proprie richieste di interruzione. Sul fronte di salita del clock, durante l'ultimo ciclo T dell'ultimo ciclo M di ogni istruzione, la CPU campiona le 2 linee di interruzione.
   Se una delle due si trova nello stato attivo (quello logico 0) la CPU riconosce la richiesta di interruzione.
   Pertanto, il tempo intercorrente tra l'invio di una richiesta di interruzione e il suo riconoscimento da parte della CPU non può mai essere superiore al tempo di esecuzione di una istruzione.
   Poichè le istruzioni durano da 4 a 23 cicli T, il ritardo nel riconoscimento dell'interruzione è compreso tra 1,6 e 9,2 microsecondi, assumendo che il clock della CPU abbia una frequenza di 2,5 MHz.

   Nota: Un problema alquanto interessante sul riconoscimento delle interruzioni da parte della CPU Z80 è cosa accade con le seguenti istruzioni "lunghe": LDIR, LDDR, CPIR, CPDR, OTIR, OTDR, INIR e INDR. Sovente, si pensa che per queste istruzioni, come per tutte le altre, le linee di interruzione sono campionate dalla CPU Z80 nell'ultimo ciclo T dell'ultimo microciclo, quando BC va a 0. Questo *non* è affatto vero. Infatti, la CPU tratta queste istruzioni come se fossero costituite da una *serie di cicli distinti di istruzioni*. Ad ogni ciclo di istruzioni corrisponde il trasferimento di un byte. Pertanto, un'interruzione può essere riconosciuta alla fine di ogni ciclo. Quando la CPU termina il servizio dell'interruzione, l'esecuzione della istruzione "lunga" continua dal punto nel quale era stata interrotta, se i contenuti dei registri utilizzati dall'istruzione non sono cambiati.

2. Le interruzioni non-mascherabili sono ammesse? In caso affermativo, quante?
   La CPU Z80 ammette un'interruzione non-mascherabile, implementata tramite un ingresso (pin 17), denominato $\overline{NMI}$, del circuito integrato a 40 pin della CPU Z80. Tale interruzione non-mascherabile consiste in un'interruzione su singola linea. Quando il pin $\overline{NMI}$ è portato nello stato attivo (ossia nello stato

logico 0), la CPU Z80 riconosce che è in corso una richiesta di interruzione non-mascherabile, e provvede al relativo servizio eseguendo un restart alla locazione 0066H. In altre parole, dopo il completamento dell'istruzione in corso al momento dell'interruzione, la CPU *inserisce logicamente* l'istruzione CALL 0066H nel programma che stata eseguendo, come prima istruzione da eseguire. In questo modo il controllo è trasferito ad un'apposita subroutine adibita alla gestione dell'interruzione non-mascherabile. Al termine di questa routine viene eseguita l'istruzione di ritorno da subroutine con la quale si restituisce il controllo al programma precedentemente interrotto. In attesa di esaminare più a fondo i particolari di questo processo nel corso degli esperimenti posti a chiusura di questo capitolo, potete sin d'ora rendervi conto del ruolo fondamentale svolto dall'operazione di chiamata di subroutine per trasferire il controllo tra il programma interrotto e la routine di gestione dell'interruzione.

3. Sono ammesse interruzioni mascherabili? In caso affermativo, quante? Lo Z80 ammette due tipi di interruzioni mascherabili, su singola linea e vettorizzate. Quelle vettorizzate sono realizzate secondo due modi detti *Modo di Interruzione 0* e *Modo di Interruzione 2*. Con la procedura di interruzione in Modo 0 il dispositivo che causa l'interruzione deve fornire una istruzione mentre, con quella in Modo 2, un indirizzo. La procedura di interruzione in Modo 1 dà luogo ad un'interruzione su singola linea con l'esecuzione automatica di un restart a partire dalla locazione 0038H. Tutti e tre tali modi di interruzione mascherabili sono implementati tramite il pin 16, denominato $\overline{INT}$, del circuito integrato della CPU Z80. Il controllo del modo di interruzione in vigore in un certo momento è affidato al software. Vale a dire che la CPU interpreta un segnale, attivo nello stato basso, inviato al suo ingresso $\overline{INT}$, come interruzione di Modo 0, 1 oppure 2 a seconda di come è stata programmata per la gestione delle interruzioni mascherabili. Il modo di interruzione è selezionato eseguendo una delle tre distinte istruzioni, IM0 (ED 46 esadecimale), IM1 (ED 56 esadecimale), ed IM2 (ED 5E esadecimale). La Tabella 6-1 riassume le interruzioni mascherabili e non-mascherabili ammesse dalla CPU Z80. Come in precedenza, ulteriori particolari saranno forniti in occasione degli esperimenti finali di questo capitolo.

4. Come fa la CPU ad invididuare il dispositivo che le richiede l'interruzione? Per quanto riguarda le interruzioni su singola linea, ai pin $\overline{NMI}$ e $\overline{INT}$ può essere collegato più di un dispositivo. Se i dispositivi collegati si riducono ad uno solo, non esiste ovviamente, alcun problema per la sua identificazione. Se tuttavia, come avviene spesso, i dispositivi che possono portare nello stato attivo uno dei due ingressi riservati alle interruzioni sono più di uno, occorrerà associare ad essi dei circuiti addizionali con cui fornire alla CPU informazioni sul dispositivo che ha generato l'interruzione. Questi circuiti supplementari presentano, di solito, la configurazione di una porta di ingresso ove a ciascun bit è associato un particolare dispositivo, potenziale sorgente di interruzione. Il dispositivo che genera l'interruzione pone il bit ad esso associato (detto FLAG ESTERNO) al valore logico 1, mentre gli altri dispositivi non richiedenti interruzione mantengono i bit ad essi relativi nello stato logico 0. La CPU può in tal modo, procedere alla lettura della porta e, esaminando i vari bit, determinare a quale routine di gestione dell'interruzione cedere il controllo. E' inoltre possibile progettare del software che risolva i casi in cui i dispositivi che richiedono l'interruzione in un dato istante siano più d'uno. Benchè, di norma, tale software si limiti a fissare un ordine di precedenza, esso può, tuttavia, assumere caratteristiche ben più sofisticate, tenendo conto del tempo trascorso dall'ultima richiesta e di quello necessario al servizio delle varie interruzioni e variando dinamicamente le priorità del sistema. Per quanto concerne l'implementazione

Tabella 6-1. Riassunto dei tipi di interruzioni della CPU Z80.

Modo	Azione	Ritorno
NMI	CALL a 0066H	RETN
IM0	Sul data bus è presente l'istruzione da eseguire	RETI
IM1	Chiamata a 0038H	RETI
IM2	Sul data bus sono presenti gli 8 bit meno significativi dell'Indirizzo della Tavola dei Vettori. Il Registro I contiene gli 8 bit più significativi Registro I da D0 a D7 Indirizzo della tavola dei vettori LO HI CALL a questo indirizzo Tavola dei Vettori	RETI

delle priorità, ricordiamo che è possibile ricorrere anche a tecniche hardware, facenti uso di diversi circuiti integrati esistenti. Il maggior pregio dell'hardware rispetto al software risiede nella più elevata velocità, mentre in quanto a flessibilità il software è di gran lunga superiore. Nel caso di interruzione vettorizzata l'istruzione (Modo 0) o l'indirizzo (Modo 2) di otto bit identifica il dispositivo. La sua funzione è, in pratica, quella di indicare alla CPU la routine di gestione dell'interruzione che deve eseguire. Per questo motivo l'istruzione compresa in un'istruzione di Modo 0 consiste abitualmente in un'istruzione di salto (BRANCH), come RST o CALL, mentre l'indirizzo fornito con una interruzione di Modo 2 non è altro che un puntatore all'indirizzo della prima istruzione della routine di gestione dell'interruzione.

5. Come può la CPU decidere sui conflitti che nascono quando due interruzioni si presentano simultaneamente?

Nel microprocessore Z80 la linea di interruzione NMI ha la precedenza rispetto alla linea $\overline{\text{INT}}$. Ne risulta che, se è vero che un'interruzione non-mascherabile può sempre interrompere la gestione di un'interruzione mascherabile, non è invece possibile il contrario (salvo eccezioni; si veda l'Esperimento 4 nel seguito di questo capitolo). Come già accennato, per gli eventuali conflitti tra due o più interruzioni non-mascherabili (oppure tra due o più interruzioni mascherabili) è necessario studiare una soluzione di tipo hardware, software o anche basata su di una combinazione mista.

6. Come viene riconosciuta una interruzione?
La CPU Z80 effettua il riconoscimento dell'interruzione eseguendo uno speciale ciclo M1, detto ciclo di riconoscimento dell'interruzione. Durante questo speciale ciclo M1, i segnali $\overline{\text{M1}}$ ed $\overline{\text{IORQ}}$ (in luogo del solito $\overline{\text{MREQ}}$) sono entrambi attivati in modo da costituire un segnale di riconoscimento dell'interruzione attivo nello stato basso, detto $\overline{\text{INTA}}$, risultante dall'operazione logica mostrata in Figura 6-2.



Figura 6-2. Segnale di riconoscimento dell'interruzione.

Nel corso degli esperimenti avremo occasione di esaminare più da vicino il ciclo di richiesta/riconoscimento di interruzione della CPU Z80.

7. Come si abilitano e disabilitano le interruzioni mascherabili?
Tanto l'abilitazione quanto la disabilitazione delle interruzioni mascherabili è effettuata in modo automatico da parte della CPU ed su controllo software. La CPU è dotata di due flip-flop interni che le permettono di conservare una traccia dell'avvenuta abilitazione o disabilitazione delle interruzioni mascherabili. Questi flip-flop sono contraddistinti dalle sigle IFF1 ed IFF2, dove IFF significa *Interrupt Flip-Flop* (flip-flop di interruzione). IFF1 è adibito alla abilitazione delle interruzioni mascherabili, mentre IFF2 serve per l'eventuale memorizzare provvisoria del contenuto di IFF1. Sia IFF1 che IFF2 sono controllati automaticamente dalla CPU in occasione delle circostanze seguenti:
a. Quando la CPU viene resettata, IFF1 ed IFF2 sono entrambi resettati così che le interruzioni mascherabili risultano disabilitate.
b. Allorchè la CPU accetta un'interruzione, IFF1 ed IFF2 sono entrambi resettati così che le istruzioni mascherabili risultano disabilitate.

Le istruzioni che la CPU Z80 deve eseguire per abilitare, oppure disabilitare, le interruzioni mascherabili sono due: DI (F3 esadecimale) per la disabilitazione (delle interruzioni) ed EI (FB esadecimale) per la abilitazione. In seguito all'istruzione EI i flip-flop IFF1 ed IFF2 sono entrambi settati, mentre sono entrambi resettati con l'esecuzione dell'istruzione DI. A proposito della istruzione EI è importante ricordare che, la CPU Z80 non accetta interruzioni mascherabili prima del termine dell'esecuzione dell'istruzione successiva a quella di EI. Ad esempio, la sequenza di istruzioni:

```
EI
RETI
```

non consente che siano accettate interruzioni mascherabili se non dopo aver eseguito l'istruzione RETI. (Si osservi che RETI è il codice mnemonico della istruzione di ritorno da un'interruzione, utilizzata nella routine di servizio delle interruzioni in luogo della più usuale istruzione RET. Su questa istruzio-

ne avremo ancora occasione di soffermarci). Il ritardo di un'istruzione, che precede l'abilitazione delle interruzioni, è fondamentale per questo motivo: le routine di gestione delle interruzioni terminano molto spesso con la sequenza di istruzioni riportata sopra. Il ritardo di un'istruzione nell'accettazione delle interruzioni garantisce che il ritorno da un'interruzione avvenga prima che ne sia accettata una nuova. Grazie a questo accorgimento si evita la crescita senza necessità dello stack in seguito ad interruzioni che intervengono mentre è ancora in corso l'esecuzione dell'istruzione EI. Si badi che l'effetto della istruzione DI è invece immediato.

Tutte le istruzioni del microprocessore Z80 relative al controllo della CPU sono riassunte nella Tabella 6-2.

Tabella 6-2. Gruppo delle istruzioni di controllo della CPU.

'NOP'	00	
'HALT'	76	
DISABILITAZIONE DELL'INT. 'DI'	F3	
ABILITAZIONE DELL'INT. 'EI'	FB	
SET MODO 0 'IM0'	ED 46	MODO 8080A
SET MODO 1 'IM1'	ED 56	CHIAMA LA POSIZIONE 0038 H
SET MODO 2 'IM2'	ED 5E	CALL INDIRETTO UTILIZZANDO COME PUNTATORE IL REGISTRO I ED 8 BIT FORNITI DAL DISPOSITIVO INTERROMPENTE

INTRODUZIONE AGLI ESPERIMENTI

Sebbene da un punto di vista concettuale sia molto semplice, la gestione delle interruzioni è di implementazione piuttosto complessa. Nelle precedenti sezioni di questo capitolo abbiamo insistito di proposito sugli aspetti teorici senza entrare nei particolari. Gli esperimenti che vi saranno ora proposti sono concepiti per esaminare da vicino proprietà inerenti alla gestione delle interruzioni nella CPU Z80, senza escludere nessun dettaglio. Proprio per questo scopo abbiamo utilizzato un folto gruppo di routine software, che mostrano esattamente ciò che avviene all'interno della CPU e della memoria allorchè una interruzione è accettata, riconosciuta e servita. Se il software controlla e visualizza lo stato di un dato sistema che evolve nel tempo allora, si dice che esegue un'*elaborazione in tempo reale*. Se i campionamenti del sistema non sono troppo rapidi rispetto alla capacità del software di avvertire e agire di conseguenza, lo stato rilevato in tempo reale del software coincide sempre con lo stato effettivo del sistema. Se, d'altro canto, i cambiamenti del sistema sono tanto rapidi che il software non è in grado di avvertirli tutti, lo stato rilevato dal software non può altro che essere impreciso. Il tempo che intercorre tra la osservazione di un fenomeno e l'esecuzione della parte di programma necessaria a visualizzare lo stato osservato assume un rilievo non trascurabile in quanto lo stato del sistema nel frattempo potrebbe essere cambiato, e di conseguenza lo stato visualizzato non coinciderebbe più con lo stato del sistema in quel preciso momento.

Per i calcolatori, anche per i più veloci, la visualizzazione dello stato in tempo reale presenta sempre un certo ritardo rispetto alla rilevazione. Mentre per alcuni sistemi, che presentano ritardi di ore, le prestazioni generali sono considerati soddisfacenti, altri vengono giudicati non abbastanza veloci a causa di un ritardo di pochi microsecondi. Il software di cui vi servirete negli esperimenti che seguono è un software in tempo reale che rileva lo stato di una CPU Z80 funzionante – impresa assai difficile, se si tiene conto di tutte le variazioni del sistema. Il tempo che intercorre tra osservazione-rapporto è, nella maggior parte dei casi, trascurabile, in quanto generalmente contenuto nell'ordine dei microsecondi. Quando il ritardo darà luogo a delle distorsioni significative, sarà nostra cura porle nella dovuta evidenza.

Gli esperimenti che sarete invitati ad effettuare sulla gestione delle interruzioni dello Z80 possono riassumersi come segue:

Esperimento N.	Commenti
1	Illustra la tecnica di interruzione su singola linea dello Z80 (Modo 1).
2	Illustra gestione delle interruzioni dello Z80 in Modo 0.
3	Illustra la gestione delle interruzioni dello Z80 in Modo 2.
4	Illustra l'interruzione non mascherabile dello Z80.
5	Tratta gli aspetti teorici dello SWITCHING del CONTESTO ed illustra due tecniche ad esso relative nell'ambito della gestione delle interruzioni dello Z80.
6	Tratta degli aspetti teorici delle interruzioni NIDIFICATE e della RIENTRANZA. E' illustrata l'organizzazione di un programma rientrante, che permette di utilizzare la medesima routine di gestione delle interruzioni da parte di più dispositivi sorgenti di interruzioni.

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è presentarvi alcuni dei particolari più notevoli della gestione dell'interruzione mascherabile dello Z80 e, soprattutto, illustrarvi praticamente la gestione dell'interruzione in Modo 1.

Programmi: INIT1, MAIN e SERV1

Codice oggetto		Codice sorgente	Commenti
3EC3	INIT1:	LD A,0C3H	; il primo byte è il codice della istruzione ; di salto
323800		LD (0038H),A	
FD216E02		LD IY,SERV1	; indirizzo della routine di
FD223900		LD (0039H),IY	; servizio # 1
ED56		IM1	; modo di interruzione 1
08		EX AF,AF'	; fissa il formato per gli spazi vuoti
3E40		LD A,40H	; per CONVDI
08		EX AF,AF'	
C3C302		JP MAIN	; salta alla routine MAIN

```
FB              MAIN:  EI                    ; abilita le interruzioni
DD210000C              LD IX,DSTACK          ; fondo dello stack dei dati
DD3600FF               LD (IX+00H),OFFH      ; temporizzatore per la visualizzazione
21E50F                 LD HL,ADDH            ; predispone il puntatore del buffer
ED57                   LD A,I                ; trova il valore di IFF2
EAD802                 JPE,HIGH
3600             LOW:  LD (HL),00H           ; valore = 0
1802                   JR NEXT
3610            HIGH:  LD (HL),10H           ; valore = 1
2B              NEXT:  DEC HL                ; sposta il puntatore del buffer
35                     DEC (HL)              ; decrementa COUNT
ED73E20F               LD (DATAL),SP         ; trascrivi SP nel buffer
21B90F                 LD HL,LEDL            ; predisponi HL per CONVDI
11E50F                 LD DE,ADDH            ; predisponi DE per CONVDI
00             DISAB:  NOP                   ; nessuna operazione
CD7CFA                 CALL CONVDI
CD09F9         DLOOP:  CALL DISPL
DD3500                 DEC (IX+00H)          ; temporizzatore per la visualizzazione
```

Codice oggetto	Codice sorgente	Commenti

```
20F8                   JR NZ,DLOOP
C3C302                 JP MAIN               ; salta indietro all'indirizzo del programma

C5             SERV1:  PUSH BC               ; salva i registri della CPU
D5                     PUSH DE
E5                     PUSH HL
F5                     PUSH AF
DDE5                   PUSH IX
FDE5                   PUSH IY               ; aggiorna il puntatore dello
DD23             DS1:  INC IX                ; stack dei dati
DD23                   INC IX
DD23                   INC IX
00                     NOP                   ; nessuna operazione
DD3600FF               LD (IX+00H),0FFH      ; predisponi il tempo di DLOOP1
DD36010A               LD (IX+01H),00AH      ; predisponi il tempo di CLOOP1
DD360202      CLOOP1:  LD (IX+02H),02H       ; predisponi il tempo di DLOOP1
21E50F                 LD HL,ADDH            ; punta al buffer del display
ED57                   LD A,I                ; trova il valore di IFF2
EA9502                 JP PE,HIGH1
3600            LOW1:  LD (HL),00H           ; valore = 0
1802                   JR NEXT1
3610           HIGH1:  LD (HL),10H           ; valore = 1
2B             NEXT1:  DEC HL                ; sposta il puntatore del buffer
34                     INC (HL)              ; incrementa ADDL
ED73E20F               LD (DATAL),SP         ; trascrivi SP nel buffer
21B90F                 LD HL,LEDL            ; predisponi HL per CONVDI
11E50F                 LD DE,ADDH            ; predisponi DE per CONVDI
CD7CFA                 CALL CONVDI
CD09F9        DLOOP1:  CALL DISPL
DD3500                 DEC (IX+00)           ; temporizzatore per la visualizzazione
20F8                   JR NZ,DLOOP1
DD3502                 DEC (IX+02)           ; temporizzatore per la visualizzazione
20F3                   JR NZ,DLOOP1
DD3501                 DEC (IX+01)           ; temporizzatore per la routine
20CD                   JR NZ,CLOOP1          ; di servizio
```

```
FDE1        POP IY      ; ripristina i registri della CPU
DDE1        POP IX
F1          POP AF
E1          POP HL
D1          POP DE
C1          POP BC
FB          EI          ; abilita le interruzioni
ED4D        RETI        ; ritorna dall'interruzione
```

Schema del circuito (Figura 6-3).

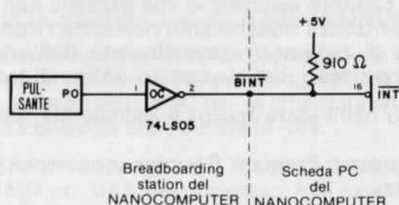


Figura 6-3. Schema N. 1: Circuiti di generazione di un segnale attivo su $\overline{INT}$.

Passo 1

Per prima cosa discutiamo su come la CPU Z80 accetta e riconosce le interruzioni mascherabili provenienti da dispositivi esterni. Come per la lettura e scrittura di I/O o in memoria è disponibile un particolare ciclo della CPU, così vi è un ciclo speciale della CPU relativo alla richiesta ed al riconoscimento di un'interruzione, detto Ciclo di Richiesta/Riconoscimento dell'interruzione. Il diagramma dei tempi di questo ciclo è rappresentato in Figura 6-4.

La CPU Z80 provvede a campionare il proprio pin $\overline{INT}$ di ingresso in corrispondenza del fronte di salita dell'ultimo microciclo di un'istruzione. Si osservi che la Figura

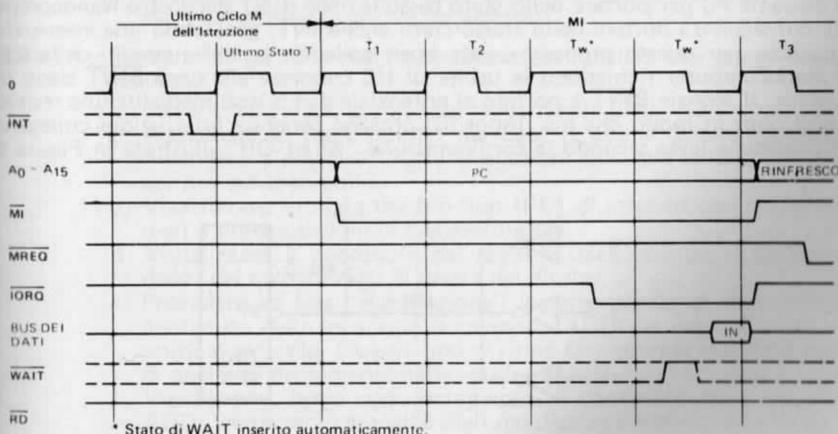


Figura 6-4. Ciclo di richiesta/riconoscimento dell'interruzione per la CPU Z80.

6-4 mostra l'ultimo ciclo M dell'istruzione. Poichè la CPU procede alla verifica del proprio stato solo in corrispondenza del fronte di salita dell'ultimo ciclo T dell'ultimo ciclo M dell'istruzione, è soltanto in corrispondenza di esso che interessa lo stato del pin $\overline{\text{INT}}$. Nel caso di segnale $\overline{\text{INT}}$ attivo (basso), la CPU, in genere, esegue un ciclo di Richiesta/Riconoscimento dell'interruzione. Il segnale $\overline{\text{INT}}$ sarà tuttavia ignorato se si verifica una delle due condizioni:

1. Il segnale $\overline{\text{BUSRQ}}$ è attivo (stato logico 0)
2. Il flip-flop IFF1 di abilitazione dell'interruzione non è al valore 1.

Il segnale $\overline{\text{BUSRQ}}$ è utilizzato dai dispositivi esterni che desiderino ottenere il controllo dei bus degli indirizzi, dei dati e di controllo. Tanto in questo come nei prossimi esperimenti di questo capitolo assumiamo che $\overline{\text{BUSRQ}}$ non sia mai attivo. Se IFF1 è al valore 0, allora le interruzioni mascherabili non sono riconosciute.

Nel corso di un ciclo di richiesta/riconoscimento dell'interruzione si verifica un certo numero di eventi importanti, qui elencati in ordine di tempo:

1. E' attivato (portato nello stato basso) il segnale $\overline{\text{M1}}$, con inizio di un apposito ciclo M1.
2. Il contenuto del registro Program Counter (contatore di programma) è posto sul bus degli indirizzi.
3. Tra T2 e T3 sono inseriti automaticamente due stati di attesa.
4. E' attivato il segnale $\overline{\text{IORQ}}$.
5. La CPU legge il dato presente sul bus dei dati dopo ulteriori stati di attesa addizionali, eventualmente inseriti.

Il trasferimento del contenuto del PC sul bus degli indirizzi dipende dal fatto che il ciclo è un ciclo M1 e non serve a nulla. Si noti l'esecuzione dell'operazione di refresh a partire da T3. Con $\overline{\text{M1}}$ ed $\overline{\text{IORQ}}$ entrambi attivi (nello stato logico 0), si ottiene l'attivazione del segnale $\overline{\text{INTA}}$, secondo quanto già detto nella sezione precedente. Questo segnale, privo d'importanza nell'elaborazione delle interruzioni in Modo 1, è invece essenziale per i dispositivi che inviano richiesta di interruzione in Modo 0 od in Modo 2 in quanto esso funge da segnale di strobe del codice di identificazione del dispositivo sul bus dei dati che la CPU dovrà successivamente leggere. La spiegazione dei due stati di attesa sarà fornita nella sezione dedicata alle interruzioni con priorità (capitolo 2).

Montare il circuito secondo lo schema. Questo circuito vi permette di servirvi del pulsante P0 per portare nello stato basso la linea $\overline{\text{BINT}}$ del vostro Nanocomputer e, di conseguenza portare nello stesso stato anche $\overline{\text{INT}}$, generando una interruzione. Si utilizza, per questo proposito, gate open collector poichè i circuiti della scheda del Nanocomputer richiedono le uscite di I/C connesse alla linea $\overline{\text{BINT}}$ siano open collector. Il segnale $\overline{\text{BINT}}$ è portato al potenziale di +5 Volt mediante una resistenza di 910 ohm in modo che più dispositivi possano generare interruzioni collegandosi alla medesima linea secondo la configurazione "wired-OR" illustrata in Figura 6-5.

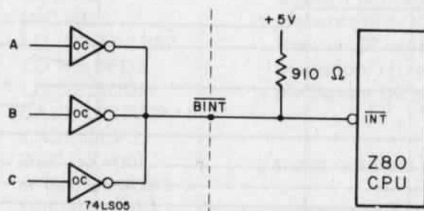


Figura 6-5. Connessione a $\overline{\text{INT}}$ in Wired-Or di più dispositivi sorgenti di interruzione.

Ciascuno dei dispositivi connessi con i fili A, B o C ha la possibilità di generare interruzioni portanto allo 0 logico l'uscita del proprio gate open collector. Poichè il resistore pull-up di 910 ohm è saldato direttamente sulla scheda del Nanocomputer, negli esperimenti con il Nanocomputer, se si vogliono generare delle interruzioni mascherabili, è necessario valersi sempre di gate open collector. Il circuito utilizzato in questo esperimento, è il circuito invertitore open collector 74LS05. Per bilanciare l'inversione del segnale si usa P0 come ingresso all'invertitore.

Passo 2

Passiamo ora all'esame delle tre routine software che userete in questo esperimento:

INIT1 Routine di inizializzazione che svolge le seguenti funzioni:

a. Carica un'istruzione di salto alla locazione 0038H.
b. Predispone nel Modo 1 il modo di interruzione dello Z80.
c. Predispone il registro A' per la chiamata alla subroutine CONVDI
d. Cede il controllo alla routine MAIN.

Nota: Nei programmi che predispongono il modo di interruzione nel Modo 0 o nel Modo 1, le istruzioni IM0 od IM1 non possono mai essere la prima istruzione. La spiegazione di questo fatto è che il comando GO inizia l'esecuzione del programma eseguendo la prima istruzione sempre in modo single-step. Poichè il single-step si vale di interruzioni di Modo 2 generate dal chip PIO, le due istruzioni precedenti non possono essere eseguite in modo single-step. Per questa ragione, nelle routine INIT0 ed INIT1, tali istruzioni non possono mai trovarsi all'inizio.

MAIN: Routine che svolge le operazioni principali. E' nel corso di questa routine che hanno luogo le interruzioni. Le sue funzioni sono:

a. Visualizzare lo stato del flip-flop di interruzione IFF1 nell'ultimo digit di sinistra del display del Nanocomputer.
b. Visualizzare il contenuto del registro stack pointer (SP) nei quattro digit di destra del display.
c. Effettuare un'"elaborazione" *decrementando* il contenuto di una locazione di memoria, all'indirizzo ADDL, di 2 unità al secondo.
d. Visualizzare il contenuto della locazione ADDL nel terzo e quarto digit del display (a partire da sinistra).

SERV1: Routine di gestione dell'interruzione. Le funzioni che svolge sono le seguenti:

a. Salvare nello stack lo stato dei registri della CPU.
b. Aggiornare lo stack pointer dei dati. (L'esame di tale funzione è rimandato a più tardi. Altre spiegazioni sulla sua validità non sono per ora indispensabili).
c. Visualizzare lo stato dei flip-flop IFF1 di interruzione nell'ultimo digit a sinistra dell'unità tastiera/display.
d. Visualizzare il contenuto del registro stack pointer in corrispondenza dei quattro digit di destra del display.
e. Procedere ad una "elaborazione" *incrementando* di dieci unità il contenuto della locazione di memoria ADDL al ritmo di circa una unità al secondo. Questo tipo di ritmo corrisponde alla metà esatta di quello di decremento nella routine MAIN.
f. Visualizzare, dopo ogni incremento, il contenuto della locazione ADDL come terzo e quarto digit del display partendo da sinistra.
g. Ripristinare lo stato dei registri della CPU, recuperandoli dallo stack.
h. Controllo di ritorno a MAIN.

Gli scambi del controllo tra queste tre routine sono riportati nel diagramma di Figura 6-6. La freccia 1 corrisponde al trasferimento del controllo alla routine MAIN in seguito al completamento di INIT1. La freccia 2 si riferisce all'acquisizione del controllo da parte di SERV1 come risposta ad una interruzione generata con il pulsante P0. Con la freccia 3 è indicata la restituzione del controllo da SERV1 a MAIN in seguito al completamento della routine di gestione dell'interruzione. Si osservi che l'istruzione utilizzata per il ritorno al termine di SERV1 è RETI. Essa è usata in luogo della solita istruzione RET per informare la CPU che si tratta di ritorno da una routine di gestione di interruzione, e non da un subroutine chiamata con una istruzione di CALL. La differenza che ne scaturisce, per il momento poco comprensibile, apparirà più evidente nella sezione dedicata alle interruzioni con priorità ed alla struttura daisy-chain nel Capitolo 9.

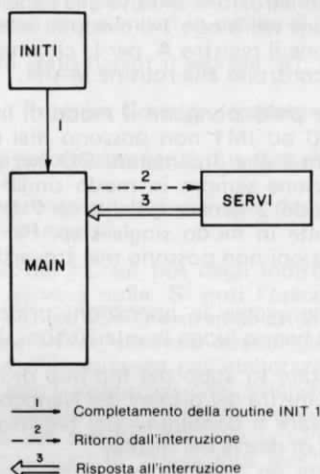


Figura 6-6. Scambio del controllo tra INIT1, MAIN e SERV1.

Per riassumere il meccanismo di interazione tra hardware e software quando viene iniziata l'esecuzione del programma a partire dalla locazione INIT1, si svolge la seguente sequenza di eventi:

1. Il modo di interruzione è predisposto in Modo 1 mediante l'esecuzione della istruzione IM1.
2. Il flip-flop IFF1 di interruzione è abilitato mediante l'esecuzione dell'istruzione EI.
3. Avviene l'attivazione di $\overline{INT}$ da parte di un dispositivo esterno.
4. La CPU riceve e riconosce l'interruzione emettendo il segnale $\overline{INTA}$ che risulta attivo nel caso che $\overline{M1}$ ed $\overline{IORQ}$ siano entrambi attivi. Contemporaneamente, il flip-flop IFF1 è resettato in modo da disabilitare ulteriori interruzioni mascherabili.
5. La CPU effettua un restart dalla locazione 0038H, trasferendo in tal modo il controllo alla routine di gestione dell'interruzione.

Iniziata l'esecuzione di INIT1. Ciò che osservate sul display del Nanocomputer dovrebbe corrispondere a quanto illustrato in Figura 6-7.

Sul display dovrebbe risultare che il flip-flop IFF1 di interruzione è abilitato (posizionato al valore logico 1), che il processo attualmente in fase di esecuzione consiste in un conto alla rovescia ad un ritmo di circa 2 unità al secondo e che il contenuto del registro stack pointer è 0F00H.

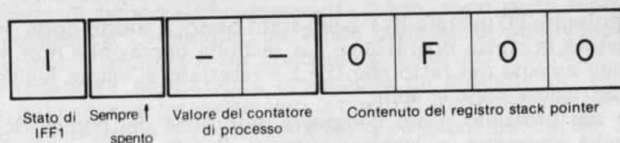


Figura 6-7. Aspetto del display e della tastiera.

Passo 3

Portate $\overline{INT}$ nello stato basso agendo sul pulsante P0. Cosa osservate?

Noi abbiamo osservato tre mutamenti sul display:

a. Il flip-flop IFF1 di interruzione è stato resettato al valore logico 0, indicando che le interruzioni sono state disabilitate. Si ricordi, infatti, che la CPU Z80, allorchè riceve e riconosce una richiesta di interruzione, disabilita automaticamente le interruzioni mascherabili.

b. Il contatore si incrementava anzichè decrementarsi. Il ritmo al quale si incrementava era circa la metà di quello precedente. Il motivo di tale variazione si spiega con il fatto che il controllo era stato trasferito momentaneamente alla routine SERV1 di servizio dell'interruzione, in risposta all'interruzione da voi generata per mezzo del pulsante.

c. Il registro stack pointer conteneva uno di questi due valori possibili:

Caso 1. 0EF2 se, al momento in cui era intervenuta l'interruzione, la routine MAIN non stava eseguendo nè l'una nè l'altra delle subroutine CONVDI e DISPL, ovvero

Caso 2. 0EEE se, al momento in cui era intervenuta l'interruzione, la routine MAIN stava eseguendo o CONVDI o DISPL.

d. Dopo 10 secondi il display aveva riassunto l'aspetto iniziale.

La probabilità che o CONVDI o DISPL fossero in fase di esecuzione al momento in cui si era verificata l'interruzione è estremamente elevata. Analizziamo, ora, i motivi per cui il contenuto del registro stack pointer è influenzato dalla posizione nel tempo dell'interruzione.

Analisi del Caso 1: La routine SERV1 accumula nello stack sei parole di due byte (ossia coppie di registri e/o registri a 16 bit). Inoltre la CPU accantona automaticamente nello stack l'indirizzo di ritorno quando esegue l'istruzione di restart alla locazione 0038H. Perciò, nel corso della gestione di un'interruzione generata da P0, sono infilate complessivamente nello stack sette coppie di due byte l'una, equivalenti a quattordici byte. L'addizione esadecimale di 0EF2 e 000E (rappresentazione esadecimale del numero decimale 14) dà come risultato 0F00.

Analisi del Caso 2: Sia CONVDI che DISPL accumulano nello stack esattamente una coppia di registri. Comprendendo l'indirizzo di ritorno previsto dalla chiamata di subroutine a CONVDI o DISPL, nonchè le sette parole di due byte accan-

tonate nello stack in seguito all'esecuzione di SERV1, sullo stack sono infilati in tutto 18 byte (12 esadecimale). Si può verificare facilmente che l'addizione esadecimale di 0EEE e 0012 dà appunto come somma 0F00.

Passo 4

Agendo sul pulsante P0 portate $\overline{INT}$ nello stato basso, e subito dopo, mentre IFF1 è resettato, ripetete la stessa operazione. La seconda operazione non mostra alcun effetto sul display a causa del fatto che IFF1 è resettato al valore logico 0, disabilitando così le interruzioni mascherabili.

L'esecuzione del prossimo passo comporta modifiche del codice di programma memorizzato nella memoria a lettura/scrittura. Potete trovare più conveniente limitarvi a seguire questo passo sul testo, rinunciando ad eseguirle praticamente.

Passo 5

Alla locazione DS1+6 nella routine SERV1 vi è un'istruzione NOP (00 esadecimale). Sostituite questa istruzione NOP con una istruzione di abilitazione delle interruzioni, EI, FB esadecimale. Iniziate l'esecuzione del programma partendo da INIT1. Agendo sul pulsante P0, attivate il segnale $\overline{INT}$. Che cosa osservate?

Ciò che noi abbiamo osservato è stato lo spegnimento totale del display. Resettate il Nanocomputer ed esaminate la memoria a partire dalla locazione DS1 +6. Noi abbiamo osservato che il codice del programma è stato distrutto. Cosa è accaduto?

Per comprendere ciò che è avvenuto, bisogna sapere che: il *segnale* $\overline{INT}$ *è sensibile al livello.* Si ricordi inoltre come la CPU Z80 campioni lo stato del pin $\overline{INT}$ in corrispondenza del fronte di salita dell'ultimo microciclo di una istruzione. Nel caso che IFF1 sia abilitato e $\overline{BUSRQ}$ non sia attivato, ha luogo il trasferimento del controllo alla routine di servizio dell'interruzione, nel nostro esempio SERV1. Il risultato è che nello stack vengono infilati un certo numero di byte. Il cambiamento che avete apportato sostituendo l'istruzione NOP con l'istruzione EI è all'origine di questo comportamento, in quanto avete creato una routine di servizio delle interruzioni in cui le interruzioni mascherabili sono abilitate troppo presto. Troppo presto in confronto alle centinaia di istruzioni durante le quali $\overline{INT}$ è basso. Ne deriva che la CPU Z80 presta il suo servizio a precchie centinaia di richieste d'interruzione prima che il pulsante ritorni nella posizione di riposo. (Se fate il conto dei tempi di esecuzione delle varie istruzioni, scoprirete che l'esecuzione dell'istruzione EI avviene dopo soli 50 microsecondi dall'istante in cui la CPU avverte il segnale $\overline{INT}$ basso). Ad ogni servizio dell'interruzione lo stack aumenta di dimensioni, espandendosi verso la parte bassa della memoria, fino a distruggere il codice del programma.

Nella versione originale della routine SERV1 il problema della sensibilità al livello non si poneva, in quanto le interruzioni mascherabili non erano abilitate sino a che la routine di servizio dell'interruzione non fosse terminata. Non sempre questa è la migliore soluzione. Una più pratica soluzione è, fare in modo che il pulsante non attivi direttamente la linea $\overline{INT}$, ma, piuttosto, funga da strobe per l'ingresso di clock edge triggered (attivato dal fronte di salita o discesa) di un flip-flop. Grazie a questa implementazione il segnale d'interruzione appare al dispositivo esterno (quale il pulsante) come se fosse un *segnale edge triggered*. La Figura 6-8 mostra come realizzare una linea di richiesta di interruzione edge triggered.

Si osservi come la transizione da basso ad alto genera in seguito a pressione del pulsante P0 posizioni l'uscita Q del flip-flop al complemento del valore logico 0 presente all'ingresso D, con conseguente attivazione del segnale $\overline{INT}$. Allorchè la

CPU riconosce l'interruzione, viene generato il segnale $\overline{INTA}$ che, a sua volta, inviato all'ingresso SD del flip-flop, determina il ritorno di Q al valore logico 0 e la disattivazione di INT. La funzionalità di questo circuito consiste proprio nella disattivazione di INT qualunque sia lo stato logico del pulsante.

I metodi che si possono usare, per avere la garanzia che INT si mantenga basso per un intervallo di tempo sufficientemente breve, sono certo molti. Il circuito di Figura 6-8 costituisce, comunque, un valido esempio di multivibrazione bistabile. Un'altra soluzione è utilizzare un condensatore ed una resistenza e realizzare un circuito multivibratore monostabile innescabile da pulsante. E' questa la tecnica che sarà illustrata quando presenteremo il circuito utilizzato per generare il segnale di richiesta di interruzione non mascherabile. La ragione per cui qui abbiamo voluto presentare il progetto di bistabile non è soltanto perchè esso assolve pienamente alla funzione richiesta, ma anche per il fatto che alcuni circuiti periferici dello Z80, quali il PIO ed il CTC, comprendono un flip-flop interno che esegue questa stessa funzione. Tale flip-flop ha il nome di *flip-flop di interruzione*. La giustificazione di tale denominazione apparirà chiara nel corso dei prossimi passi.

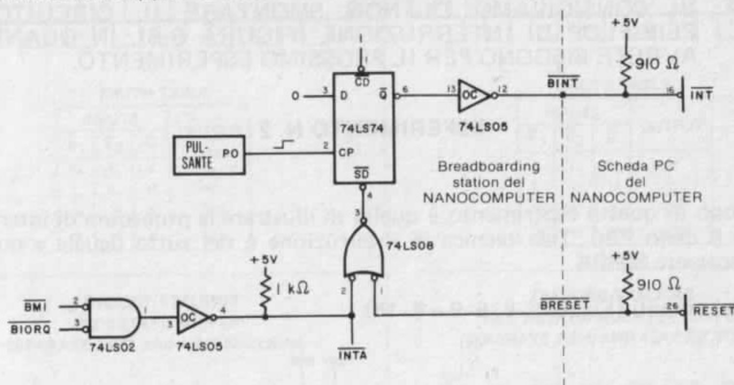


Figura 6-8. Circuiti di un flip-flop di interruzione pendente (in attesa).

Passo 6

Montate il circuito con il flip-flop di interruzione pendente (in attesa) secondo lo schema di Figura 6-8. Ripristinate il contenuto della memoria del computer com'era prima dell'espansione incontrollata dello stack che avete appena finito di osservare. Caricate, perciò, di nuovo i programmi e cambiate in EI, FB esadecimale, l'istruzione NOP alla locazione DS1+6. Eseguite il programma iniziando da INIT1 e premete il pulsante P0 così che la linea collegata all'ingresso di clock del flip-flop presenta una transizione dal valore logico 0 a quello 1, mantenendosi di poi al valore logico 1. Che cosa osservate?

Noi abbiamo osservato che è stata generata esattamente un'interruzione.

Passo 7

Cambiate di nuovo in NOP l'istruzione EI in DS1+6. Iniziate l'esecuzione del programma partendo da INIT1. Agite sul pulsante. Mentre è in corso di esecuzione la routine di servizio dell'interruzione, agite sul pulsante ancora una volta. Cosa osservate?

Noi abbiamo osservato che due interruzioni sono state servite, l'una immediatamente di seguito all'altra, senza notare nessun ritorno alla routine MAIN. Il flip-flop (Figura 6-8) ha memorizzato la seconda richiesta di interruzione sino al momento in cui la istruzione EI, posta al termine della routine SERV1, non ha indotto la CPU ad accettarla e riconoscerla. Il secondo $\overline{INTA}$, naturalmente, provvede a resettare il flip-flop. La ragione del nome di questo flip-flop si riferisce, dunque, alla sua funzione di memorizzare un'interruzione in sospeso.

Passo 8

Premete il pulsante. Mentre è in corso di esecuzione la routine di servizio dell'interruzione, premete ancora il pulsante a più riprese. Cosa osservate?

Noi abbiamo osservato che le interruzioni servite una di seguito all'altra sono, di nuovo, solamente due. La ragione di ciò è che il flip-flop può memorizzare una sola interruzione pendente (in attesa).

NOTA: VI CONSIGLIAMO DI NON SMONTARE IL CIRCUITO DEL FLIP-FLOP DI INTERRUZIONE (FIGURA 6-8), IN QUANTO NE AVRETE BISOGNO PER IL PROSSIMO ESPERIMENTO.

ESPERIMENTO N. 2

Scopo

Lo scopo di questo esperimento è quello di illustrare la procedura di interruzione in Modo 0 dello Z80. Tale tecnica di interruzione è del tutto uguale a quella del microprocessore 8080A.

Schemi dei circuiti (Figure 6-8, 6-9 e 6-10).

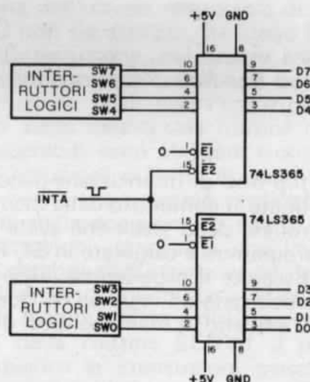


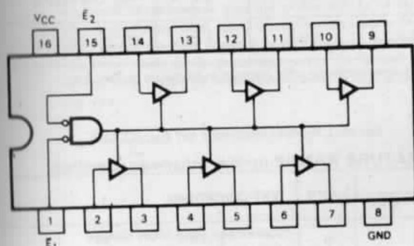
Figura 6-9. Schema N. 2.

Oltre al circuito riportato in Figura 6-8, per questo esperimento vi servirete anche del circuito raffigurato in Figura 6-9.

Configurazioni dei pin dei circuiti integrati (Tavole 6-1, 6-2 e 6-3)

Tavola 6-1. Caratteristiche del 74LS365.

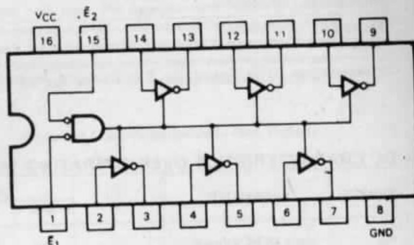
T54LS365/T74LS365
HEX 3-STATE BUFFER WITH COMMON 2-INPUT NOR ENABLE



TRUTH TABLE

INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	L
L	L	H	H
H	X	X	(Z)
X	H	X	(Z)

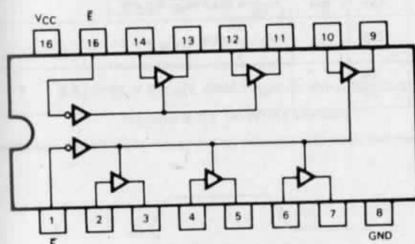
T54LS366/T74LS366
HEX 3-STATE INVERTER BUFFER WITH COMMON 2-INPUT NOR ENABLE



TRUTH TABLE

INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	H
L	L	H	L
H	X	X	(Z)
X	H	X	(Z)

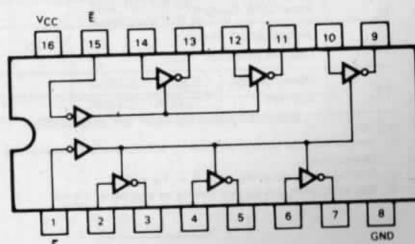
T54LS367/T74LS367
HEX 3-STATE BUFFER SEPARATE 2-BIT AND 4-BIT SECTIONS



TRUTH TABLE

INPUTS		OUTPUT
$\bar{E}$	D	
L	L	L
L	H	H
H	X	(Z)

T54LS368/T74LS368
HEX 3-STATE INVERTER BUFFER SEPARATE 2-BIT AND 4-BIT SECTIONS



TRUTH TABLE

INPUTS		OUTPUT
$\bar{E}$	D	
L	L	H
L	H	L
H	X	(Z)

DESCRIPTION – The LS365/366/367/368 are high speed hex buffers with 3-state outputs. They are organized as single 6-bit or 2-bit/4-bit, with inverting or non-inverting data (D) paths. The outputs are designed to drive 15 TTL Unit Loads or 60 Low Power Schottky loads when the Enable ($\bar{E}$) is LOW.

When the Output Enable Input ($\bar{E}$) is HIGH, the outputs are forced to a high impedance "off" state. If the outputs of the 3-state devices are tied together, all but one device must be in the high impedance state to avoid high currents that would exceed the maximum ratings. Designers should ensure that Output Enable signals to 3-state devices whose outputs are tied together are designed so there is no overlap.

Tavola 6-1. Caratteristiche del 74LS365 (seguito).

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS365AX T54LS366AX T54LS367AX T54LS368AX	4.5 V	5.0 V	5.5 V	-55° C to +125° C
T74LS365AX T74LS366AX T74LS367AX T74LS368AX	4.75 V	5.0 V	5.25 V	0° C to +70° C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS	
			MIN	TYP	MAX			
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.4	3.4			I_{OH} = −1.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.4	3.1			I_{OH} = −2.6 mA	
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA	
I_{OZH}	Output Off Current HIGH				20	μA	V_{CC} = MAX, V_{OUT} = 2.4 V, $V_{\bar{E}}$ = 2.0 V	
I_{OZL}	Output Off Current LOW				−20	μA	V_{CC} = MAX, V_{OUT} = 0.4 V, $V_{\bar{E}}$ = 2.0 V	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−40		−225	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current	LS365A·LS367A		13.5	24	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 4.5 V	
		LS366A·LS368A		11.8	21			

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25° C.
3. Not more than one output should be shorted at a time.

AC CHARACTERISTICS: T_A = 25°C, V_{CC} = 5.0 V (See Page 100 for Waveforms)

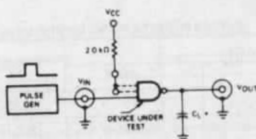
SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS	
		MIN	TYP	MAX			
t_{PLH} t_{PHL}	Propagation Delay, Data to Output LS365A · LS367A			10 16	ns	Fig. 2	C_L = 45 pF
t_{PLH} t_{PHL}	Propagation Delay, Data to Output LS366A · LS368A			10 16	ns	Fig. 1	C_L = 45 pF
t_{PZH}	Output Enable Time to HIGH Level			16	ns	Figs. 4, 5	C_L = 45 pF
t_{PZL}	Output Enable Time to LOW Level			30	ns	Figs. 3, 5	R_L = 667 Ω
t_{PLZ}	Output Disable Time from LOW Level			15	ns	Figs. 3, 5	C_L = 5.0 pF
t_{PHZ}	Output Disable Time from HIGH Level			23	ns	Figs. 4, 5	R_L = 667 Ω

Tavola 6-1. Caratteristiche del 74LS365 (seguito).

AC TEST CIRCUITS AND WAVEFORMS

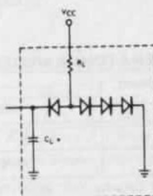
The following test circuits and conditions represent SGS-ATES's typical AC test procedures. The output loading for standard Low Power Schottky devices is a 15 pF capacitor. Experimental evidence shows that test results using the additional diode-resistor load are within 0.2 ns of the capacitor only load. The capacitor only load also has the advantage of repeatable, easily correlated test results. The input pulse rise and fall times are specified at 6 ns to closely approximate the Low Power Schottky output transitions through the active threshold region. The specified propagation delay limits can be guaranteed with a 15 ns input rise time on all parameters except those requiring narrow pulse widths. Any frequency measurement over 15 MHz or pulse width less than 30 ns must be performed with a 6 ns input rise time.

Test Circuit for Standard Output Devices

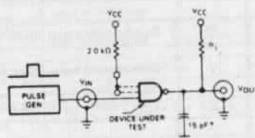


*Includes all probe and jig capacitance

Optional Load (Guaranteed—Not Tested)



Test Circuit for Open Collector Output Devices

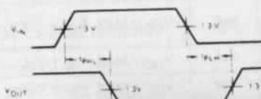


*Includes all probe and jig capacitance

Pulse Generator Settings (unless otherwise specified)

Frequency = 1 mHz
Duty Cycle = 50%
t_{TLH} (t_r) = 6 ns
t_{THL} (t_f) = 6 ns
Amplitude = 0 to 3 V

Waveform for Inverting Outputs



Waveform for Non-inverting Outputs

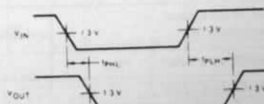
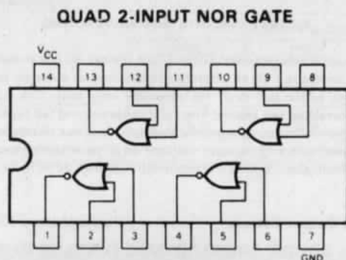


Tavola 6-2. Caratteristiche del 74LS02.

QUAD 2-INPUT NOR GATE



GUARANTEED OPERATING RANGES

PART.NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS02X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS02X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−20		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.6	3.2	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			2.4	5.4	mA	V_{CC} = MAX, Inputs Open

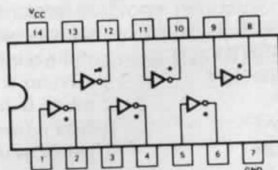
AC CHARACTERISTICS: T_A = 25°C (See Page 273 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tavola 6-3. Caratteristiche del 74LS05.

HEX INVERTER



*OPEN COLLECTOR OUTPUTS

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS05X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS05X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
I_{OH}	Output HIGH Current				100	μA	V_{CC} = MIN, V_{OH} = 5.5 V, V_{IN} = V_{IL}
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{CCH}	Supply Current HIGH			1.2	2.4	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			3.6	6.6	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C **(See Page 273 for Waveforms)**

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output		14	22	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output		10	18	ns	C_L = 15 pF, R_L = 2.0 kΩ

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.

Programmi INITO

In aggiunta ai due programmi MAIN e SERV1 già elencati a proposito dell'Esperimento N. 1, ci si varrà del seguente programma:

Codice oggetto		Codice sorgente	Commenti
3EC3	INITO:	LD A,0C3H	; il primo byte è il codice della istruzione
			; di salto
323800		LD (0038H),A	; caricalo alla locazione RST
FD216E02		LD IY,SERV1	; indirizzo della routine di
FD223900		LD (0039H),IY	; servizio # 1
ED46		IM0	; modo 0 di interruzione
08		EX AF,AF'	; fissa il formato per gli spazi vuoti
3E40		LD A,40H	; per CONVDI
08		EX AF,AF'	
C3C302		JP MAIN	; salta alla routine MAIN

Passo 1

Esaminiamo per prima cosa il Modo 0 di interruzione dello Z80, confrontandolo con il Modo 1. Si può subito notare come per entrambi i modi di interruzione, il ciclo di richiesta/riconoscimento delle interruzioni sia esattamente lo stesso. La differenza più importante tra i due modi è il significato dell'identificatore fornito dal dispositivo che causa l'interruzione. Nel Modo 1 di interruzione il dispositivo si limita a fornire un segnale di richiesta di interruzione, attivo nello stato basso. Nella procedura di interruzione in Modo 0, invece, oltre ad un segnale di richiesta di interruzione, la CPU attende dal dispositivo un byte di istruzione ad otto bit, che consiste in una istruzione ad uno o più byte. I circuiti che utilizzerete per questo esperimento non sono in grado di fornire istruzioni con più di un byte. La CPU gestisce l'interruzione nel Modo 0 leggendo, decodificando ed eseguendo l'istruzione fornita dal dispositivo che ha richiesto l'interruzione. Le istruzioni più frequentemente utilizzate sono una delle 8 istruzioni ad un solo byte RST, poichè presentano sufficientemente flessibilità e, come tutte le istruzioni di chiamata a subroutine salvano automaticamente il contenuto del registro PC nello stack. Si osservi che, se il dispositivo fornisce il byte FF, codice operativo dell'istruzione RST 38H, si ottiene lo stesso risultato di una interruzione in Modo 1.

Nello schema N. 2 in Figura 6-9 è riportato il circuito che utilizzerete unitamente al pulsante P0 in Figura 6-8 per i vostri esperimenti sulle interruzioni in Modo 0 dello Z80. Il pulsante fornisce alla CPU il segnale di richiesta di interruzione, mentre il circuito dello Schema N. 2 pone sul bus dei dati un'istruzione ad un byte in un insieme ben definito del ciclo di richiesta/riconoscimento dell'interruzione. Si osservi che il segnale $\overline{INTA}$ serve a porre sul bus dei dati il dato impostato con gli interruttori logici. Quando il segnale $\overline{INTA}$ è disattivato, i buffer fanno ritorno al loro stato ad alta impedenza.

Eseguite i collegamenti circuitali descritti nello Schema N. 2, se non lo avete già fatto per l'Esperimento N. 1, collegate il pulsante P0 al flip-flop di interruzione pendente (in attesa) come illustrato in Figura 6-8. Procedete al montaggio del circuito di Figura 6-10, che ha la funzione di indicare lo stato della linea $\overline{HALT}$. Questo

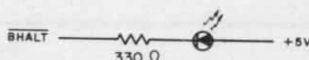


Figura 6-10. Circuito indicatore dello stato del pin $\overline{HALT}$ della CPU Z80.

circuito, anche se non serve per generare interruzioni di Modo 0, si rivelerà utile più avanti nel corso di questo esperimento.

Passo 2

Il software utilizzato da questo esperimento consiste in tre routine INIT0, MAIN e SERV1. Per la descrizione delle funzioni espletate da MAIN e SERV1, si rimanda all'Esperimento N. 1 di questo stesso capitolo. INIT0 è del tutto uguale alla routine INIT1 di cui si è fornita la documentazione nell'Esperimento N. 1, con la differenza che programma la CPU Z80 per gestire le interruzioni in Modo 0 anzichè Modo 1.

I trasferimenti del controllo tra le routine INIT0, SERV1 e MAIN sono schematizzati in Figura 6-11.

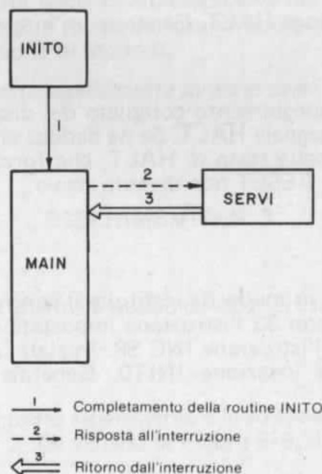


Figura 6-11. Trasferimento del controllo tra le routine INIT0, MAIN e SERV1.

Ponete tutti gli interruttori nella posizione ON, così da impostare il numero FF esadecimale. L'istruzione di codice FF consiste in un salto a subroutine alla locazione 0038. Iniziate l'esecuzione del programma a partire dalla locazione INIT0. Osservate come il display si presenta come nel corso dell'Esperimento N. 1.

Passo 3

Generate un'interruzione agendo sul pulsante P0. L'effetto dovrebbe essere lo stesso di quello osservato durante lo Esperimento N. 1, quando avete generato interruzioni di Modo 1.

Nota: Le istruzioni IM0, oppure IM1, non devono mai trovarsi all'inizio del programma. La ragione di questa affermazione sta nel fatto che il comando GO inizia l'esecuzione del programma eseguendo la prima istruzione in single-step. Poichè l'esecuzione in single-step necessita di interruzioni di Modo 2 generate dal chip PIO, è chiaro che le due istruzioni IM0 e IM1 non possono essere eseguite in single-step. E' questo il motivo per il quale, nelle routine INIT0 ed INIT1, queste due istruzioni NON compaiono mai per prime.

A questo punto dovete apportare alla routine MAIN la variazione che segue:

Locazione	*Contenuto*	*Da cambiare in*
DISAB	00	F3 (disabilita le interruzioni)

Con questo cambiamento il flip-flop IFF1 di interruzione viene disabilitato, e rimane tale finchè viene eseguita di nuovo l'istruzione EI, memorizzata alla locazione MAIN. Lo scopo è di non gestire nessuna interruzione durante l'esecuzione di una delle due routine CONVDI e DISPL. Il Modo 0 di interruzione dello Z80 ci permette di porre sul bus dei dati delle istruzioni, come ad esempio, incrementa lo stack pointer, e di eseguirle. Per questa ragione non possiamo permettere che siano accettate delle interruzioni mentre è in corso l'esecuzione di una subroutine di MAIN.

Passo 4

Cambiate da FF in 76 l'istruzione impostata dagli interruttori. 76 corrisponde al codice operativo dell'istruzione HALT. Generate un'interruzione. Cosa osservate?

Noi abbiamo osservato lo spegnimento completo del display e l'accensione del LED che visualizza lo stato del segnale $\overline{HALT}$. Se ne deduce che lo Z80 ha eseguito l'istruzione HALT e si è posto nello stato di HALT, che forza la CPU ad eseguire operazioni NOP purchè il segnale RESET non diventa attivo.

Passo 5

Premere il tasto RESET in modo da restituire il controllo al sistema operativo del Nanocomputer. Sostituite con 33 l'istruzione impostata con gli interruttori. Questo è il codice esadecimale dell'istruzione INC SP. Iniziate, di nuovo, la esecuzione del programma a partire dalla locazione INITO. Generate alcune interruzioni. Cosa osservate?

Noi abbiamo osservato che il contenuto del registro stack pointer, visualizzato nei quattro digit di destra del display del Nanocomputer, ha subito un incremento pari al numero di interruzioni generate.

Passo 6

Cambiate in 3B l'istruzione impostata. 3B corrisponde al codice operativo dell'istruzione DEC SP. Cosa prevedete di vedere, in questo caso, generando delle interruzioni?

Noi abbiamo osservato che il contenuto del registro stack pointer è stato decrementato tante volte quante sono state le interruzioni generate. In questo passo, come nel precedente Passo 4, è parso che il contatore continuasse a contare, senza interruzione. Il flip-flop di interruzione, IFF1, è stato soggetto a delle brevi transizioni dallo stato logico 1 a quello 0, coincidenti con il rilevamento ed il riconoscimento di una interruzione, per tornare, di nuovo, allo stato logico 1 in occasione dell'esecuzione dell'istruzione EI alla locazione MAIN. Poichè, per passare dal byte 33 al byte 3B, è sufficiente spostare un solo interruttore, e precisamente di SW3, voi potete facilmente incrementare e decrementare il valore dello stack pointer, spostando, semplicemente l'interruttore SW3 e generando, contemporaneamente, delle interruzioni.

Passo 7

Sostituite l'istruzione impostata con gli interruttori con C5, codice operativo di PUSH BC. Generate una interruzione. Cosa osservate?

Noi abbiamo osservato un decremento di due del contenuto dello stack pointer. Questo non ci deve sorprendere in quanto i byte posti sullo stack dall'istruzione PUSH BC sono due.

Passo 8

Cambiate l'istruzione in NOP, codice esadecimale 00. In questo caso, le interruzioni interessano solamente il contenuto, visualizzato sul display, di IFF1, che passa in un istante dal valore logico 1 a quello 0 e torna quindi a 1. Giunti a questo punto, dovreste essere in condizione di scegliere una qualsiasi altra istruzione di un byte e prevedere esattamente ciò che apparirà sul display in occasione di una interruzione, con gli interruttori predisposti in modo da fornire quella data istruzione nel corso del riconoscimento dell'interruzione di Modo 0.

Nota: I circuiti relativi a questo esperimento saranno usati anche per quello successivo.

ESPERIMENTO N. 3

Scopo

Lo scopo di questo esperimento è quello di illustrare la gestione delle interruzioni nel Modo 2 dello Z80.

Schema del circuito

Il circuito utilizzato per questo esperimento è lo stesso di quello già utilizzato in occasione dell'Esperimento N. 2. (Si vedano le Figure 6-8, 6-9 e 6-10).

Programmi INIT2, SERV2 e SERV3

In aggiunta alle routine MAIN e SERV1, questo esperimento richiede i seguenti programmi:

Codice oggetto		Codice sorgente	Commenti
ED5E	INIT2:	IM2	; modo di interruzione 2
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; setta il registro delle interruzioni
FD216E02		LD IY,SERV1	; prima routine di servizio
FD22000F		LD (TABLE),IY	; metti il suo indirizzo nella tabella dei ; vettori
FD21F502		LD IY,SERV2	; seconda routine di servizi
FD22020F		LD (TABLE+2),IY	; metti il suo indirizzo nella tabella dei ; vettori
FD216B03		LD IY,SERV3	; terza routine di servizio
FD22040F		LD (TABLE+4),IY	; metti il suo indirizzo nella tabella dei ; vettori

08	EX AF,AF'	; fissa il formato per CONVDI
3E40	LD A,40H	
08	EX AF,AF'	
C3C302	JP MAIN	; salta alla routine MAIN

Codice oggetto	Codice sorgente	Commenti
76	SERV2: HALT	; arresta il Nanocomputer

Codice oggetto	Codice sorgente	Commenti
C5	SERV3: PUSH BC	; salva i registri della CPU
D5	PUSH DE	
E5	PUSH HL	
F5	PUSH AF	
DDE5	PUSH IX	
FDE5	PUSH IY	
DD23	DS3: INC IX	; aggiorna il puntatore dello
DD23	INC IX	; stack dei dati
DD23	INC IX	
00	NOP	; nessuna operazione
DD3600FF	LD (IX+00H),0FFH	; predisposizione del tempo di DLOOP3
DD36010A	LD (IX+01H),00AH	; predisposizione del tempo di CLOOP3
DD360202	CLOOP3: LD (IX+02H),02H	; predisposizione del tempo di DLOOP3
21E50F	LD HL,ADDH	; punta al buffer del display
ED57	LD A,I	; cerca il valore di IFF2
EA9203	JP PE,HIGH3	
3600	LOW3: LD (HL),00H	; valore = 0
1802	JR NEXT3	
3610	HIGH3: LD (HL),10H	; valore = 1
2B	NEXT3: DEC HL	; sposta il puntatore del buffer
34	INC (HL)	; incrementa ADDL
34	INC (HL)	; incrementa ADDL
ED73E20F	LD (DATAL),SP	; trascrivi SP nel buffer
21B90F	LD HL,LEDL	; predisponi HL per CONVDI
11E50F	LD DE,ADDH	; predisponi DE per CONVDI
CD7CFA	CALL CONVDI	
CD09F9	DLOOP3: CALL DISPL	
DD3500	DEC (IX+00)	; temporizzatore per la visualizzazione
20F8	JR NZ,DLOOP3	
DD3502	DEC (IX+02)	; temporizzatore per la visualizzazione
20F3	JR NZ,DLOOP3	
DD3501	DEC (IX+01)	; temporizzatore per la routine
20CC	JR NZ,CLOOP3	; di servizio
FDE1	POP IY	
DDE1	POP IX	; ripristina i registri della CPU
F1	POP AF	
E1	POP HL	; abilita le interruzioni
D1	POP DE	
C1	POP BC	
FB	EI	
ED4D	RETI	; ritorna dall'interruzione

Passo 1

Se non lo avete già fatto, montate subito i circuiti che utilizzerete in questo esperimento.

Tra i modi di gestione delle interruzioni disponibili sullo Z80, il Modo 2 è indubbiamente quello più potente. Mentre l'hardware necessario alla sua implementazione è esattamente lo stesso di quello occorrente per il Modo 0, il software associato presenta alcune differenze. L'eguaglianza dell'hardware dipende dal fatto che, anche in Modo 2 la CPU si aspetta dal dispositivo che ha generato l'interrupt un codice di otto bit. Questo codice di otto bit, tuttavia, non è trattato come codice operativo di un'istruzione, come nel Modo 0, ma come il byte meno significativo di un indirizzo di memoria di 16 bit.

L'indirizzo completo è ricavato combinando il contenuto del registro I con il byte fornito dal dispositivo, come illustrato in Figura 6-12, in modo da formare un indirizzo di sedici bit, detto *indirizzo della tabella dei vettori*.



Figura 6-12. Formazione di un indirizzo della tabella dei vettori.

Si noti che tutti i chip periferici dello Z80 e, in particolare, i chip CTC e PIO, richiedono che il bit meno significativo di questo indirizzo abbia il valore logico 0. Pertanto, è conveniente far corrispondere, tutti gli indirizzi delle interruzioni di Modo 2 a locazioni di memoria pari, predisponendo, ad esempio, l'hardware ed il software per indirizzi della tabella dei vettori e byte dai dispositivi periferici sempre *pari*.

L'indirizzo a sedici bit formato dal registro I e dal byte di identificazione è interpretato dalla CPU Z80 (in Modo 2) come un puntatore alla locazione di memoria contenente l'indirizzo della routine di gestione dell'interruzione. Consideriamo il seguente esempio. Se il contenuto del registro I fosse 10 ed il dispositivo richiedente l'interruzione dovesse fornire il byte di identificazione 32, nelle locazioni 1032 e 1033 dovrebbe essere memorizzato il byte di ordine, rispettivamente, inferiore e superiore dell'indirizzo della routine di gestione dell'interruzione. In particolare, se il contenuto di queste due locazioni di memoria fosse il seguente:

Locazione di memoria	**Contenuto**
1032	70
1033	02

Allora, risulterebbe che l'indirizzo della prima istruzione della routine di gestione dell'interruzione sarebbe 0270. La CPU effettuerebbe, in tal caso, una chiamata a subroutine all'indirizzo 0270, salvando nello stack il contenuto del registro PC. La routine di gestione dell'interruzione dovrebbe terminare con l'istruzione RETI, che restituisce il controllo alla procedura interrotta.

In molti esempi di gestione di interruzione dello Z80 in Modo 2 il registro I è caricato all'inizializzazione con un dato valore XY e non viene più modificato. Questo valore è un puntatore ad un blocco di memoria di 256 byte contigui, costituito dalle locazioni comprese tra XY00 ed XYFF. Questo blocco di memoria è chiamato *tabella dei vettori di interruzione* in quanto le sue locazioni contengono gli indirizzi di inizio delle routine di gestione delle interruzioni. Dal momento che ciascuna coppia, XY00-XY01, XY02-XY03, . . ., XYFE-XYFF, contiene, almeno in potenza l'indirizzo di una routine di gestione di interruzione, per ogni valore del registro I, sono possibili 128 interruzioni distinte in Modo 2. Inoltre, se 128 non fossero abbastanza, vi sarebbe sempre la possibilità di cambiare il registro I! Per la maggior parte delle applicazioni questo numero di interruzione risulta più che sufficiente. Per questi fatti, relativamente alla struttura delle interruzioni, la CPU Z80, è come uno dei microprocessori più potenti. Il prezzo di questa flessibilità e potenza è 256 byte di memoria indirizzabile. Si tenga presente che, poichè il ciclo di richiesta/riconoscimento delle interruzioni non cambia da un modo di interruzione all'altro, il tempo di risposta per le interruzioni di Modo 2 ha la stessa durata di quello relativo al tipo di interruzione più semplice, ossia il Modo 1.

Passo 2

Il software utilizzato in questo esperimento comprende tre nuove routine: INIT2, SERV2 e SERV3. Soffermiamoci un momento ad esaminarle sommariamente:

INIT2: Routine di inizializzazione che esegue le funzioni seguenti:

1. Programma il modo di interruzione in Modo 2, eseguendo ad esempio l'istruzione IM2.
2. Costruisce la Tabella dei Vettori di Interruzione con gli indirizzi di partenza delle routine di gestione delle interruzioni SERV1, SERV2, SERV3. La Tabella 6-3 illustra come è fatta la Tabella dei Vettori di Interruzione.

Tabella 6-3. Tabella dei vettori di interruzione costruita dalla routine INIT2.

Locazione di memoria	Contenuto
TABLE	Byte di ordine inferiore dell'indirizzo SERV1
TABLE + 1	Byte di ordine superiore dell'indirizzo SERV1
TABLE + 2	Byte di ordine inferiore dell'indirizzo SERV2
TABLE + 3	Byte di ordine superiore dell'indirizzo SERV2
TABLE + 4	Byte di ordine inferiore dell'indirizzo SERV3
TABLE + 5	Byte di ordine superiore dell'indirizzo SERV3

Nota: Il registro I è caricato con la metà più significativa dell'indirizzo della Tabella che, unitamente agli indirizzi SERV1, SERV2 e SERV3, è riportato nella Tabella A-1 dell'Appendice A.

3. Inizializza il registro I, eseguendo, ad esempio, l'istruzione LD I,A.
4. Abilita le interruzioni mascherabili, eseguendo la istruzione EI

SERV2: Routine consistente nell'unica istruzione — HALT.

SERV3: Routine che dà luogo all'ormai familiare visualizzazione del contenuto di IFF1, del contatore e del registro stack pointer. La routine di servizio delle interruzioni SERV3, incrementa il contatore di due unità per

volta per circa dieci secondi, in luogo dei conteggi in avanti (SERV1) oppure indietro (MAIN) con incremento di una unità, visti negli esempi precedenti.

Lo scambio del controllo tra INIT2, MAIN, SERV1, SERV2 e SERV3 è riprodotto in forma schematica in Figura 6-13.

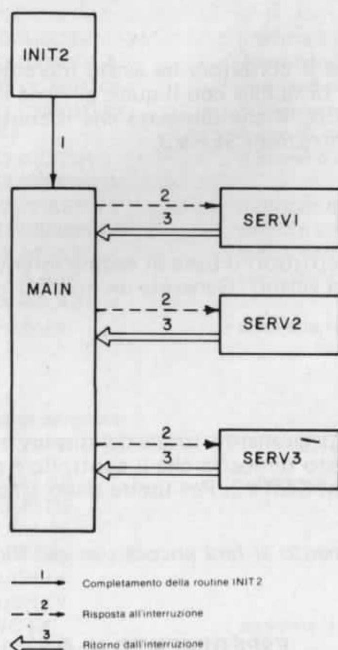


Figura 6-13. Trasferimento del controllo tra INIT2, MAIN, SERV1, SERV2 e SERV3.

Mantenete il led indicatore collegato al segnale $\overline{\text{BHALT}}$ in modo da avere una segnalazione quando la CPU Z80 entra nello stato HALT. Il relativo circuito è già stato illustrato nella Figura 6-10.

Passo 3

Iniziate l'esecuzione del programma partendo dalla locazione INIT2. Predisponete gli interruttori in modo da corrispondere alla metà meno significativa dell'indirizzo di SERV1 nella tabella dei vettori. (Si consulti la Tabella dei Simboli Principali riportata in Appendice A oppure si dia un'occhiata al codice oggetto della routine INIT2). Generate un'interruzione. Cosa osservate?

Noi abbiamo osservato l'esecuzione della routine di servizio delle interruzioni. Ce ne siamo resi conto dal comportamento del contatore. Esso ha infatti subito un incremento di 10 ad un ritmo pari alla metà di quello con il quale MAIN decrementa il contatore.

Passo 4

Impostate con gli interruttori il byte di ordine dell'indirizzo della routine SERV3 nella tabella dei vettori. Generate un'interruzione. Cosa osservate?

Noi abbiamo osservato che il contatore ha avuto incrementi di due unità per volta ad un ritmo pari alla metà di quello con il quale avviene il decremento del contatore da parte della routine MAIN, il che dimostra che il controllo è stato trasferito alla routine di servizio delle interruzioni SERV3.

Passo 5

Programmate con gli interruttori il byte di ordine inferiore dell'indirizzo della routine SERV2 nella tabella dei vettori. Generate un'interruzione. Cosa osservate?

Noi abbiamo osservato lo spegnimento totale del display e l'accensione dell'indicatore luminoso di HALT. Questo dimostra che il controllo è stato trasferito alla routine di gestione delle interruzioni SERV2. Per uscire dallo stato HALT potete premere il tasto RESET.

Nota: Nel prossimo esperimento si farà ancora uso del flip-flop di interruzione pendente (in attesa).

ESPERIMENTO N. 4

Scopo

Questo esperimento ha lo scopo di illustrare praticamente la tecnica di interruzione non mascherabile dello Z80.

Schema del circuito (Figura 6-14)

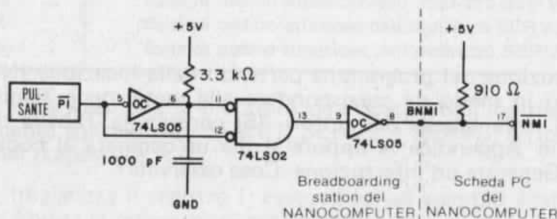


Figura 6-14. Schema N. 3.

Oltre al circuito di Figura 6-8, per questo esperimento è necessario il circuito di Figura 6-14.

Programmi INIT1N e SERVN

Oltre alla routine MAIN e SERV1, nel corso di questo esperimento utilizzerà il seguente software.

Codice oggetto		Codice sorgente	Commenti
3EC3	INIT1N:	LD A,0C3H	; il primo è una istruzione byte di salto
326600		LD (0066H),A	; interruzione non-mascherabile
FD211903		LD IY,SERVN	; indirizzo per il servizio
FD226700		LD (0067H),IY	; di interruzioni non-mascherabili
ED56		IM1	; modo di interruzione 1
3EC3		LD A,0C3H	; il primo è una istruzione byte di salto
323800		LD (0038H),A	
FD216E02		LD IY,SERV1	; indirizzo della routine di
FD223900		LD (0039H),IY	; servizio # 1
08		EX AF,AF'	; fissa il formato per gli spazi
3E40		LD A,40H	; per CONVDI
08		EX AF,AF'	
C3C302		JP MAIN	; salta alla routine MAIN

Codice oggetto		Codice sorgente	Commenti
C5	SERVN:	PUSH BC	; salva i registri della CPU
D5		PUSH DE	
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
DD23	DSN:	INC IX	; aggiorna il puntatore dello stack dei dati
DD23		INC IX	
DD23		INC IX	
00		NOP	; nessuna operazione
DD3600FF		LD (IX+00H),00FH	; predisposizione del tempo di DLOOPN
DD36010A		LD (IX+01H),00AH	; predisposizione del tempo di CLOOPN
DD360202	CLOOPN:	LD (IX+02H),02H	; predisposizione del tempo di DLOOPN
21E50F		LD HL,ADDH	; punta al buffer del display
ED57		LD A,I	; cerca il valore di IFF2
EA4003		JP PE,HIGHN	
3600	LOWN:	LD (HL),00H	; valore = 0
1802		JR NEXTN	
3610	HIGHN:	LD (HL),10H	; valore = 1
ED73E20F	NEXTN:	LD (DATAL),SP	; trascrivi SP nel buffer
21B90F		LD HL,LEDL	; predisponi HL per CONVDI
11E50F		LD DE,ADDH	; predisponi DE per CONVDI
CD7CFA		CALL CONVDI	
CD09F9	DLOOPN:	CALL DISPL	
DD3500		DEC (IX+00)	; temporizzatore per la visualizzazione
20F8		JR NZ,DLOOPN	
DD3502		DEC (IX+02)	; temporizzatore per la visualizzazione
30F3		JR NZ,DLOOPN	
DD3501		DEC (IX+01)	; temporizzatore per la routine di servizio ; della interruzione

```
20CF          JR NZ,CLOOPN
FDE1          POP IY          ; ripristina i registri della CPU
DDE1          POP IX
F1            POP AF
E1            POP HL
D1            POP DE
C1            POP BC
ED45          RETN            ; ritorna dall'interruzione non-mascherabile
```

Passo 1

La tecnica di interruzione non-mascherabile si differenzia in modo sostanziale da quella di interruzione mascherabile discussa negli Esperimenti 1, 2 e 3:

1. Per generare un interruzione non-mascherabile si utilizza un differente ingresso della CPU, chiamato NMI.
2. A differenza delle interruzioni mascherabili, per le quali la risposta della CPU Z80 avviene sotto il controllo del software (EI, DI, IM0, IM1, IM2), una interruzione non-mascherabile non potrà mai essere ignorata dalla CPU e sarà sempre interpretata allo stesso modo. Questo è dovuto all'esistenza di un solo modo di interruzione non-mascherabile.
3. L'interruzione non-mascherabile ($\overline{NMI}$) è una interruzione a singola linea. Un dispositivo che avanzi richiesta di interruzione non-mascherabile non deve far altro che attivare il segnale NMI. Non è, quindi, necessario nessun byte di identificazione, del tipo di quelli utilizzati per i Modi 0 e 2 di interruzione mascherabile. La richiesta di interruzione non-mascherabile (NMI) viene servita dalla CPU eseguendo un restart alla locazione esadecimale 0066.
4. Le temporizzazioni relative alla richiesta ed al riconoscimento di una interruzione non-mascherabile sono differenti da quelle delle interruzioni mascherabili. La Figura 6-15 illustra le temporizzazioni relative all'operazione di richiesta di una interruzione non-mascherabile.

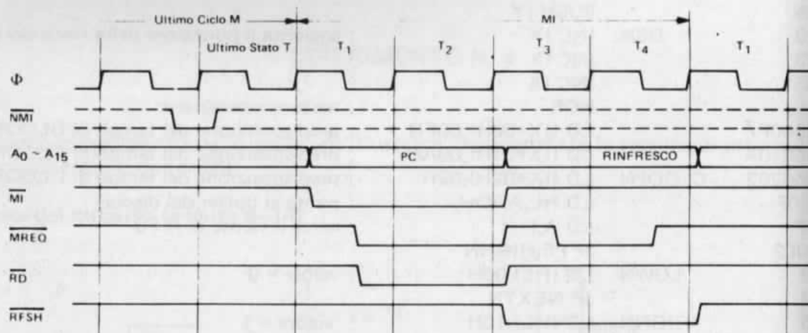


Figura 6-15. Operazione di richiesta di interruzione non-mascherabile.

5. Il campionamento del pin $\overline{NMI}$ da parte della CPU avviene in corrispondenza del fronte di salita dell'ultimo ciclo T compreso nell'ultimo ciclo M di una istruzione. Se il segnale NMI è attivo (stato logico 0), viene eseguito un normale ciclo di "fetch" del codice operativo, con attivazione dei segnali $\overline{M1}$, $\overline{MREQ}$ e $\overline{RD}$. Una conseguenza importante di questo fatto è che si esegue, oltre al "fetch", pure una operazione di refresh della memoria. Si badi che

non è generato alcun segnale di riconoscimento della interruzione (come il NOR logico di $\overline{\text{M1}}$ ed $\overline{\text{IORQ}}$ a proposito delle interruzioni mascherabili) in quanto superfluo. Il dispositivo che ha generato l'interrupt sarà infatti, servito comunque senza bisogno di sincronizzazioni varie per trasferire il byte di identificazione. Durante il ciclo M1, la CPU ignora il contenuto del bus dei dati e infila nello stack il contenuto del registro PC.

6. Poichè l'interruzione non-mascherabile ha la più alta priorità, il servizio di una interruzione mascherabile può essere sospeso in qualsiasi momento allo scopo di servire una interruzione non-mascherabile. Di segnali a priorità più elevata di quella del segnale $\overline{\text{NMI}}$ ne esiste uno solo. Il segnale $\overline{\text{BUSRQ}}$, con il quale si chiede alla CPU di vedere il controllo dei bus dei dati, degli indirizzi e di controllo ad un'altra CPU o ad un dispositivo, ha, infatti, il sopravvento sul segnale $\overline{\text{NMI}}$.
7. Se la CPU accetta un'interruzione non-mascherabile, le interruzioni mascherabili sono automaticamente disabilitate. Tale risultato è ottenuto resettando semplicemente il flip-flop IFF1. Al completamento del servizio dello NMI è opportuno ripristinare IFF1 al valore che presentava al momento dell'interruzione non-mascherabile. Pertanto, durante l'esecuzione dell'istruzione RETN si copia in IFF1 il contenuto di IFF2, che è rimasto inalterato durante la richiesta e il servizio del NMI. L'istruzione RETN, (ED45 esadecimale) è una particolare istruzione che va utilizzata solamente nelle routine di servizio dello NMI per ripristinare il flip-flop IFF1 allo stato presente all'arrivo del NMI.

Vale la pena di osservare come lo Z80 non preveda nessuna istruzione per esaminare il contenuto di IFF1, mentre è invece possibile l'esame di IFF2 mediante una delle due istruzioni LD A,I o LD A,R che copiano IFF2 nel flag P/V. Pertanto il digit più a sinistra del display visualizza in realtà lo stato di IFF2, e non quello di IFF1. Dal momento che, nel caso di interruzioni mascherabili, IFF1 ed IFF2 sono sempre uguali, non ci sono stati problemi negli Esperimenti da 1 a 3. Inoltre, poichè il servizio delle interruzioni non-mascherabili agisce sullo stato di IFF1, e non di IFF2, sarete in grado, nel corso di questo esperimento, di comprendere le differenze tra i due flip-flop.

Passo 2

Siete invitati a montare il circuito illustrato in Figura 6-8 e 6-14. Il segnale NMI dello Z80 è definito, nei vari manuali tecnici, come un segnale *edge triggered*. Come potete osservare nel circuito riprodotto qui sopra, si utilizza un resistore di 3,3 kilohm e un condensatore di 1000 picofarad per ritardare l'arrivo di un 1 logico all'ingresso 11 del gate NOR 74LS02. Il risultato è un impulso negativo della durata di circa 500 nanosecondi sull'uscita 13 del gate NOR. Le Figure 6-15 e 6-16 mostrano le temporizzazioni dei segnali più importanti di questo circuito. Il motivo per il quale abbiamo generato un segnale $\overline{\text{NMI}}$ di così piccola durata, è che, in base ai nostri esperimenti, l'impulso $\overline{\text{NMI}}$ deve mantenersi attivo per un tempo compreso tra soli 80 e 500 nanosecondi.

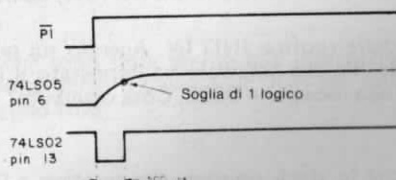


Figura 6-16. Generazione di un segnale NMI di 500 ns.

Passo 3

Oltre alle routine MAIN e SERV1, per questo esperimento utilizzeremo due nuove routine, INIT1N e SERVN.

INIT1N: Routine di inizializzazione che esegue le seguenti funzioni:
1. Carica nelle locazioni 0066H-0068H l'istruzione di salto a SERVN.
2. Predispone in Modo 1 il modo di interruzione mascherabile.
3. Carica nelle locazioni 0038H e 003AH l'istruzione di salto SERV1.
4. Predispone i registri e la memoria per la chiamata della subroutine CONVDI.

SERVN: Routine di gestione delle interruzioni non-mascherabili, che esegue le seguenti funzioni:
1. Salva lo stato della CPU.
2. Ferma il contatore per 10 secondi.
3. Visualizza il contatore.

I trasferimenti del controllo tra INIT1N, MAIN, SERV1 e SERVN sono schematizzati in Figura 6-17.

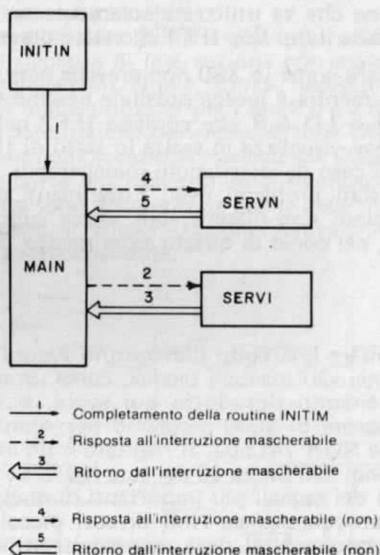


Figura 6-17. Trasferimenti del controllo tra INIT1N, MAIN, SERV1 e SERVN.

Iniziate l'esecuzione delle routine INIT1N. Agendo sul pulsante P0, generate una interruzione mascherabile. Poichè software ha impostato il Modo di interruzione 1, la CPU esegue un restart alla locazione 0038. Cosa osservate?

Noi abbiamo osservato che lo stack pointer, che puntava a 0F00H ed è stato abbassato a 0EEEH. Questo dimostra che è stata eseguita la routine di gestione delle interruzioni SERV1.

Passo 4

Allo scopo di dimostrare che la CPU Z80 non può mai ignorare una interruzione non-mascherabile, resettate il Nanocomputer e cambiate in NOP (00 esadecimale) la prima istruzione EI della routine MAIN. Si osservi che, solo alla penultima istruzione di SERV1 viene seguita l'istruzione EI. Pertanto durante il servizio di una interruzione mascherabile, le interruzioni sono disabilitate.

Iniziate l'esecuzione a partire dalla locazione INIT1N. Notate che entrambi i flip-flop IFF1 ed IFF2 si trovano nello stato logico 0, confermando che le interruzioni mascherabili sono disabilitate. Provate a generare una interruzione mascherabile agendo sul pulsante P0. Questo vostro tentativo non dovrebbe produrre nessun effetto, in quanto prosegue normalmente l'esecuzione della routine MAIN. Questo significa che non è stata generata alcuna interruzione mascherabile, che la CPU cioè, ignora i segnali sul suo ingresso $\overline{\text{INT}}$. Azionate il pulsante $\overline{\text{P1}}$ in modo da generare una richiesta di NM1. Cosa osservate?

Noi abbiamo osservato che il controllo è passato alla routine SERVN che mantiene fermo il contatore per 10 secondi. Abbiamo altresì osservato che lo stack pointer è stato abbassato a 0EEEH. Ne deduciamo che la gestione dell'interruzione non-mascherabile ha avuto luogo. Ne consegue che il riconoscimento del segnale NMI non risente del fatto che le interruzioni mascherabili sono disabilitate.

Passo 5

Ripristinare l'istruzione EI alla locazione MAIN, sostituendo cioè, 00 con FB. Generate un'interruzione mascherabile mediante il pulsante P0. Durante l'esecuzione della routine di servizio dell'interruzione, agite su P1 generando un'interruzione non-mascherabile. Cosa osservate?

Noi abbiamo osservato che, nonostante il flip-flop IFF2 sia al valore logico 0, (interruzioni disabilitate) abbiamo potuto generare lo stesso una interruzione non-mascherabile. Questa affermazione è ampiamente dimostrata dal fatto che il contatore di SERV1 viene bloccato per circa 10 secondi dalla routine SERVN di servizio dello NMI.

Passo 6

Agendo sul pulsante $\overline{\text{P1}}$, generate un'interruzione non-mascherabile. Mentre è ancora in corso il servizio di questa interruzione, agite di nuovo su $\overline{\text{P1}}$ per generare una seconda interruzione non-mascherabile. Cosa osservate?

Noi abbiamo osservato che sono state servite due interruzioni non-mascherabili. Lo stack pointer è stato spinto verso il basso a 0EDCH per circa 10 secondi mentre era in corso il servizio del primo NMI.

Passo 7

Generate una interruzione non-mascherabile servendovi del pulsante $\overline{\text{P1}}$ e prendete nota del valore visualizzato sul primo digit a sinistra. Agite su P0 nell'intento di

generare un'interruzione mascherabile. Cosa osservate?

Noi abbiamo osservato che il contenuto del primo digit a sette segmenti sulla destra era un 1 logico e che, nonostante ciò, non siamo riusciti a generare un'interruzione mascherabile. Questo non contraddice la regola che, quando IFF1 si trova nello stato logico 1, le interruzioni mascherabili sono abilitate? La nostra risposta è negativa per il fatto che il primo digit a sinistra sul display rispecchia il contenuto di IFF2, e non quello di IFF1. Si ricordi, al proposito, che durante il servizio di un'interruzione non-mascherabile il contenuto di IFF2 corrisponde allo stato che aveva IFF1 all'occorenza del NMI. Ricordate che quello che appare sul display è il contenuto di IFF2, in quanto non vi è alcun modo di conoscere il contenuto di IFF1. Se ne deduce che, dal momento che la CPU non ha riconosciuto delle interruzioni mascherabili, IFF1 si doveva trovare nello stato logico 0, mentre il display riportava lo stato logico 1 di IFF2. Poichè, al verificarsi di un'interruzione non-mascherabile la CPU resetta immediatamente IFF1, tutto questo corrisponde perfettamente con quanto ci aspettavamo.

Si osservi che l'interruzione mascherabile resta memorizzata nel flip-flop di interruzione pendente (in attesa) e, pertanto, servita alla fine del servizio del NMI.

Passo 8

Generate un'interruzione non-mascherabile. Mentre è ancora in corso il servizio di questa interruzione, provate a generare un'interruzione mascherabile. Come ben sapete, la CPU non accetterà immediatamente questa seconda interruzione, che rimane memorizzata nel flip-flop di interruzione pendente (in attesa). A questo punto, mentre è sempre in corso la gestione dell'interruzione non-mascherabile, generate una *seconda* interruzione non-mascherabile. Prestate particolare attenzione all'*ordine secondo il quale avviene la gestione di queste due interruzioni in sospeso.* Cosa osservate?

Noi siamo rimasti piuttosto sorpresi nell'osservare che è stata servita l'interruzione mascherabile prima della seconda interruzione non-mascherabile. Al riguardo non abbiamo nessuna spiegazione da dare. Si tratta semplicemente di una particolarità della CPU Z80.

Nota: Il circuito per la generazione di richieste NMI (Figura 6-14) è utilizzato ancora nel prossimo esperimento. Il circuito flip-flop di interruzione, come il generatore di impulsi $\overline{NMI}$, sarà utilizzato nell'Esperimento N. 6.

ESPERIMENTO N. 5

Scopo

In questo esperimento sarà presentato il concetto di *switching del contesto* e saranno altresì presentate due tecniche che ricorrono ad esso durante la gestione di interruzioni con lo Z80.

Schema del circuito

Per questo esperimento vi servirete del circuito già utilizzato per generare interruzioni non-mascherabili nell'Esperimento N. 4, riportato nella Figura 6-14.

Programmi

Saranno utilizzate le routine: INIT1N, MAIN e SERVN (Si veda l'Esperimento N. 4).

Passo 1

Nel corso di questo esperimento analizzeremo le prime nove e le ultime sette istruzioni della routine SERVN:

```
PUSH  BC
PUSH  DE
PUSH  HL
PUSH  AF
PUSH  IX
PUSH  IY
•  •  •
POP  IY
POP  IX
POP  AF
POP  HL
POP  DE
POP  BC
RETN
```

Le istruzioni PUSH e POP salvano e ripristinano, rispettivamente, lo stato della CPU. Lo scopo di tali istruzioni consiste nel prelevare il *contesto o ambiente fisico* nel quale si trova la CPU al momento dell'interruzione. La conservazione del *contesto* si ottiene memorizzando tutti i dati del programma interrotto in una zona di memoria inaccessibile alla routine di servizio dell'interruzione, che non potrà, pertanto, modificarli. Una volta completato il servizio della interruzione, tutti i dati sono riportati nel posto che occupavano al momento dell'interruzione. Per quanto riguarda il programma interrotto, i dati si ritrovano così al giusto posto, e inoltre, essendo stato memorizzato quale istruzione era stata interrotta, l'esecuzione del programma può riprendere dal punto giusto. La procedura che stiamo esaminando è detta *switching* del *contesto*; in base ad esso il controllo di una CPU è trasferito da un processo o funzione (rappresentante il programma interrotto, MAIN) ad un altro processo o funzione (rappresentante la routine di servizio dell'interruzione, SERVN). Lo *switching del contesto* è definito come l'operazione di salvataggio dell'ambiente fisico del programma interrotto in modo che la sua esecuzione possa essere ripresa regolarmente, una volta gestita l'interruzione.

L'onere dello switching del contesto può essere a carico o della routine di servizio delle interruzioni o della CPU stessa se tale operazione è svolta automaticamente durante il ciclo di riconoscimento delle interruzioni.

Molti minicalcolatori ed alcuni microcomputer procedono automaticamente al salvataggio in memoria di tutti i registri della CPU, servendosi di propri circuiti interni, particolarmente veloci, prima di trasferire il controllo alla routine di gestione dell'interruzione. Il pregio principale di questa tecnica è proprio la sua velocità. Il suo difetto è, invece, quello di salvare spesso più registri di quanti non siano necessari, impegnando una area di memoria più vasta di quella necessaria.

Lo switching del contesto con lo Z80 deve, invece, essere eseguito dalla routine di gestione dell'interruzione. La routine SERVN utilizza le istruzioni PUSH e POP per salvare lo stato iniziale della CPU nello stack. Dal momento che SERVN utilizza tutti i registri del banco principale della CPU, è necessario salvarli tutti. Se, invece, la routine utilizzasse solo alcuni dei registri della CPU, non ci sarebbe alcuna necessità di salvare anche quelli che la routine non usa. Vediamo ora di calcolare il tempo occorrente per lo switching del contesto effettuato da SERVN.

Istruzione	Numero di cicli T
PUSH AF	11
PUSH BC	11
PUSH DE	11
PUSH HL	11
PUSH IX	15
PUSH IY	15
POP AF	10
POP BC	10
POP DE	10
POP HL	10
POP IX	14
POP IY	14
Totale di cicli T	142

A 2,5 MHz, ossia con cicli T della durata 400 ns circa, per uno switch del contesto la routine SERV1 impiega 56,8 microsecondi. Se per molte applicazioni questa velocità è più che sufficiente ve ne sono altre che richiedono uno switching del contesto ancor più rapido.

Lo Z80 dispone di un'alternativa allo switching illustrato che utilizza le istruzioni:

EXX: tempo di esecuzione pari a 4 cicli T
EX AF,AF': tempo di esecuzione pari a 4 cicli T

Queste due istruzioni determinano lo scambio dei contenuti dei registri A, B, C, D, E, H, L ed F con quelli corrispondenti A', B', C', D', E', H', L' ed F' in 8 cicli T, ossia 1,6 microsecondi alla frequenza di clock di 2,5 MHz. Ne deriva che queste due istruzioni permettono di effettuare uno switch del contesto, comprendente pure il salvataggio ed il recupero di IX ed IY dello stack, in

$$(2 \times 1{,}6) + (15 + 14) \times 2 \times 0{,}4 = 17{,}6 \text{ microsecondi.}$$

Anche se questo può senz'altro considerarsi un notevole progresso, siamo tuttavia ancora lontani dalla velocità dei circuiti di switching del contesto automatico del quale sono dotati alcuni Mini- e Microcalcolatori.

L'impiego delle istruzioni EXX ed EX AF, AF', è limitata da due importanti restrizioni. ESSE NON POSSONO ESSERE UTILIZZATE PER GESTIRE INTERRUZIONI NIDIFICATE E DISTRUGGONO I CONTENUTI DEI REGISTRI DEL GRUPPO ALTERNATIVO.
Allorchè tratteremo delle istruzioni nidificate, in occasione del prossimo esperimento vi renderete conto di come esse diano luogo a scambi più volte ripetuti, per cui i dati salvati la volta precedente vanno perduti in quella successiva. Una trattazione più completa delle interruzioni nidificate è rimandata all'Esperimento N. 6.

Se non si è già provveduto, montate il circuito per interruzione non-mascherabile con i collegamenti illustrati nello schema N. 3 di Figura 6-14. Caricate le routine INIT1N, MAIN e SERVN.

Passo 2

Iniziate l'esecuzione del programma INIT1N. Generate una interruzione non-mascherabile. Ciò che dovreste osservare è l'interruzione dell'esecuzione di MAIN, deducibile dal fatto che SERVN è l'unica routine che mantiene costante il contenuto del contatore per circa 10 secondi. Una volta completato il servizio dell'interruzione non-mascherabile, MAIN riprende il conteggio dal punto esatto in cui l'aveva lasciato. Sostituite le relative istruzioni PUSH e POP con le istruzioni di scambio: EXX ed EX AF, AF', ponendo NOP in corrispondenza dei byte rimasti inutilizzati. Iniziate

l'esecuzione di INIT1N. Cosa osservate?

Noi abbiamo osservato un display parzialmente distrutto per circa 10 secondi seguito da un'altra visualizzazione caotica. Se avessimo generato un'altra interruzione non-mascherabile, quanto apparso sul display sarebbe cambiato e rimasto costante per 10 secondi, dopo di che sarebbe cambiato di nuovo. Questo comportamento dimostra che l'effetto dell'interruzione si è prodotto puntualmente ma che il display non funzionava a dovere. Dove si trova l'errore? In questo esperimento abbiamo preferito fornirvi uno spunto, lasciando a voi il compito di sistemare il display.
SUGGERIMENTO: Guardate a cosa serve il registro A' nella routine CONVDI!

Nota: ***Il prossimo esperimento utilizza lo stesso circuito che avete impiegato ora.***

ESPERIMENTO N. 6

Scopo

In questo esperimento ci proponiamo di trattare i concetti di *interruzioni nidificate e di programma rientrante*. Tutte le routine di gestione delle interruzioni, delle quali ci siamo valsi in questo capitolo, sono rientranti. Ci occuperemo di cosa significa l'"essere rientrante", del significato di "rientranza" e delle condizioni da un programma deve soddisfare per essere rientrante.

Schema del circuito (Figura 6-18)

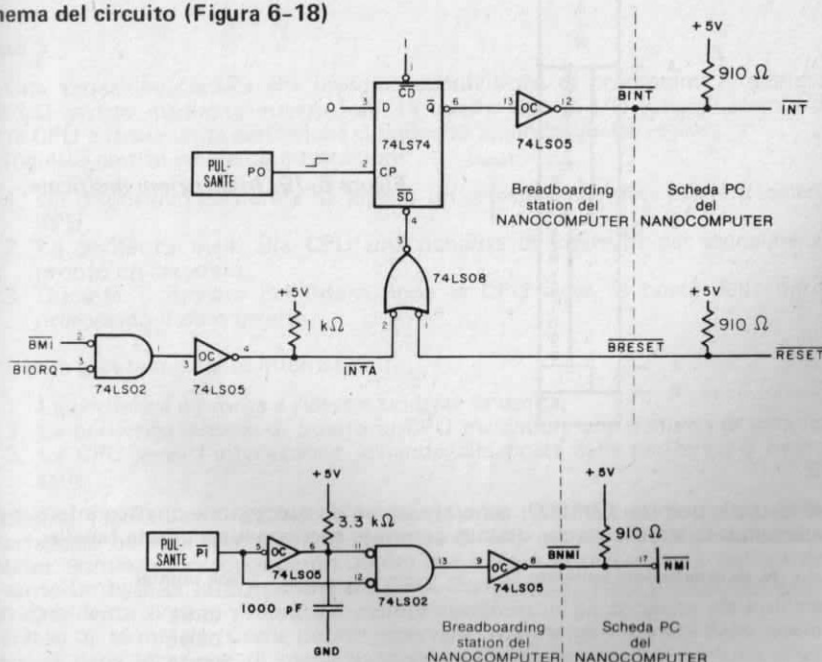


Figura 6-18. Schema N. 4.

Si noti che entrambi questi circuiti sono già serviti per gli esperimenti precedenti.

Programmi

In questo esperimento utilizzeremo routine INIT1N, MAIN, SERV1 e SERVN.

Passo 1

Iniziate l'esecuzione da INIT1N. Generate una interruzione mascherabile con il pulsante P0. Mentre questa interruzione viene gestita, con il pulsate P1 generate una interruzione non-mascherabile. Cosa osservate?

Noi abbiamo osservato come prima cosa il trasferimento del controllo a SERV1 che ha cominciato a incrementare il contatore. Non appena è stata generata l'interruzione non-mascherabile, il controllo è stato ceduto a SERVN, che ha bloccato il contatore per circa 10 secondi, trascorsi i quali il contatore ha cominciato ad incrementarsi sino al raggiungimento di un totale di 10 incrementi. Alla fine il controllo è stato restituito alla routine MAIN che decrementa il contatore.

Quanto accaduto si riassume dicendo che l'interruzione non-mascherabile era *annidata* all'interno della routine di gestione della interruzione mascherabile. La Figura 6-19 illustra quanto avete appena finito di osservare; l'interruzione No. 1 è quella mascherabile e la No. 2 quella non-mascherabile.

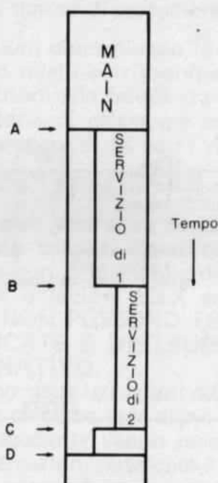


Figura 6-19. Interruzioni nidificate.

Passo 2

Con lo stack pointer a 0F000, generate in rapida successione quattro interruzioni non-mascherabili. Verificate che quanto osservate corrisponda a questa tabella:

N. di interruzioni annidate	Stack pointer
0	0EEE
1	0EDC
2	0ECA
3	0EB8
4	0EA6

La tabella sta a significare che, in assenza di interruzioni, il registro stack pointer ha il contenuto 0F00. In presenza di una interruzione la sommità dello stack si trova alla locazione 0EEE. In presenza di 2 interruzioni (1 interruzione annidata nell'altra), la sommità dello stack è arrivata a 0EDC. Con 4 interruzioni (3 nidificate) lo stack pointer ha come contenuto 0EB8. Infine la quarta interruzione porta la sommità dello stack a 0EA6. Per il trasferimento dall'uno all'altro livello di nidificazione si impiegano complessivamente 10 secondi, la durata, cioè, di un esecuzione di SERVN. Per la ultima interruzione in ordine di tempo, l'intervallo di 10 secondi è tutto intero, mentre, per le altre interruzioni, è spezzato in due poichè parte dell'intervallo è trascorso prima dell'arrivo dell'interruzione successiva e la restante parte immediatamente dopo il completamento del suo servizio. Più precisamente, considerando l'interruzione n, se l'interruzione n+1 si è verificata x secondi dopo l'interruzione n, 10-x secondi della routine di servizio dell'interruzione n sono trascorsi dopo il completamento della routine di servizio della n+1—esima interruzione.

Quello che avete appena finito di osservare sono interruzioni nidificate sempre dello *stesso tipo*, e precisamente interruzioni non-mascherabili. Ciascuna interruzione può essere considerata come un processo distinto che esegue un programma sempre uguale, e precisamente SERVN; dopo tre interruzioni, SERVN si trovava in esecuzione contemporaneamente per tre processi, e così via. Dopo un certo numero di interruzioni, vi erano in corso altrettante esecuzioni di SERVN, tutte giunte a differenti stadi di esecuzione. In questo caso l'ordine secondo il quale le varie esecuzioni sono completate corrisponde ad un algoritmo particolare della nidificazione in base al quale l'ultimo processo (in questo caso interruzione) è completato per primo (come in un registro Last In, First Out – primo a entrare, ultimo a uscire –).

Passo 3

Una situazione classica che presenta condivisione di programmi è rappresentata dall'I/O gestito mediante interruzioni. In questo tipo di I/O tutti gli scambi di dati tra la CPU e le sue unità periferiche si svolgono secondo queste regole:

Ingresso gestito mediante interruzioni:

1. Un dispositivo periferico ha pronto un carattere che deve essere trasmesso alla CPU.
2. La periferica invia alla CPU una richiesta di interrupt per segnalare che ha pronto un carattere.
3. Durante il servizio dell'interruzione la CPU legge la porta della periferica, prelevando il dato pronto.

Uscita gestita mediante interruzioni:

1. La periferica è pronta a ricevere un byte in uscita.
2. La periferica avverte di questo la CPU inviandole una richiesta di interruzione.
3. La CPU serve l'interruzione inviando alla porta della periferica il byte necessario.

Facciamo l'ipotesi che la CPU sia collegata a più terminali del tutto uguali tra loro interfacciati ad essa secondo una struttura di I/O gestita mediante interruzioni. A ciascun terminale corrisponde un utente che invierà attraverso una tastiera di tanto in tanto un byte di informazione alla CPU. Ogni battuta serve prelevando il carattere corrispondente al tasto premuto e memorizzandolo in un apposita zona di memoria riservato al terminale. Come potete osservare, tramite gli indirizzi della porta di ingresso e della locazione di memoria destinata a memorizzare il byte pervenuto dal terminale, la routine di gestione dell'interruzione è sempre la stessa per tutti i terminali. Le tecniche di gestire questa situazione sono due.

Tecnica N. 1: Copie Multiple

Con questa tecnica bisogna memorizzare per ognuno dei terminali una distinta routine di gestione dell'interruzione. Ciascuno dei terminali fornirà un suo codice di identificazione al momento stesso in cui richiede un'interruzione. A seconda del codice id (identificazione) il controllo sarà indirizzato all'opportuna copia della routine di servizio dell'interruzione. Le copie differiranno tra di loro per due byte soli, il codice del dispositivo nell'istruzione IN ed il byte low dell'indirizzo della locazione di memoria ove è memorizzato il carattere pervenuto.

Tecnica N. 2: Condivisione del Codice

Tutti i terminali utilizzano la medesima routine di gestione delle interruzioni in un modo molto simile a quello del Passo 2 di questo esperimento, allorchè la stessa interruzione era nidificata all'interno di se stessa. In questa routine di gestione delle interruzioni di più dispositivi bisogna includere alcune importanti funzioni che non sarebbero necessarie nella tecnica a copie multiple:

a. Una logica apposita che associ il codice id del terminale con l'indirizzo della porta di I/O e della locazione dove memorizzare il carattere in arrivo.
b. Una logica che permetta che l'esecuzione della routine di servizio delle interruzioni possa essere interrotta da interruzioni dello stesso tipo, che sia contemporaneamente in svolgimento più di una esecuzione, pur garantendo a ciascuna esecuzione l'integrità del corrispondente ambiente. La routine di gestione delle interruzioni deve, cioè, essere ***rientrante.***

Per garantire che un programma sia rientrante sono sufficienti queste tre condizioni:

1. Il codice non deve modificare sé stesso.
2. Lo stato dei registri della CPU deve essere salvato all'ingresso nella routine e ripristinato subito dopo l'uscita dalla stessa.
3. Gli spazi riservati ai dati associati ai diversi utenti della routine devono essere mutuamente esclusivi.

Tutte le routine di servizio delle interruzioni che avete utilizzato nel corso di questi esperimenti sono rientranti. Nel passo che segue esamineremo sotto questo aspetto una di esse, e precisamente SERVN.

Passo 4

Controlliamo la routine SERVN per verificare se tutte e tre le condizioni precedenti sono soddisfatte. E' innanzitutto evidente che SERVN non modifica se stessa. Per quanto concerne il salvataggio dello stato della CPU, questo è esattamente quanto fanno le istruzioni PUSH e POP che si trovano al principio e alla fine della routine. Per verificare la condizione che gli spazi riservati ai dati non si sovrappongano, focalizziamo la nostra attenzione sul modo d'impiego del registro IX. Si osservi che IX è utilizzato come puntatore in memoria. Le tre locazioni di memoria IXX+00, IX+01 ed IX+02 sono tutti byte di temporizzazione, utilizzati dalla routine SERVN per mantenere fermo il display per circa 10 secondi. Pertanto, se l'esecuzione della routine SERVN dovesse essere interrotta, non importa se da se stessa oppure da un'altra routine che utilizzi lo spazio riservato ai dati, sarebbe essenziale conservare i valori contenuti in queste tre locazioni di memoria in modo che, quando viene ripresa l'esecuzione di SERVN, il contatore sul display se ne stia sempre complessivamente fermo per 10 secondi. Se la routine che provoca l'interruzione dovesse distruggere il contenuto di tali locazioni, i byte di temporizzazione sarebbero alterati con prevedibili conseguenze sul tempo di fermata del contatore sul display. Questo tempo potrebbe diventare interminabile o brevissimo, a seconda di cosa la routine di

gestione dell'interruzione lascia nelle locazioni IX+00, IX+01 ed IX+02. La condizione che gli spazi riservati ai dati siano distinti è l'unico modo per risolvere il problema di conservare i dati del programma.

Poichè qui si tratta di locazioni di memoria, non è applicabile la solita tecnica dello switching del contesto descritta nel corso dell'esperimento precedente. Un possibile sistema è quello di salvare nello stack tutti i byte di temporizzazione, caricandoli nei registri della CPU ed eseguendo poi le opportune istruzioni PUSH. Prima di ripassare il controllo a SERVN, utilizzando l'istruzione POP si ricaricano in memoria i byte salvati. Questa tecnica, accettabile quando si tratta con pochi byte di dati, diventa troppo onerosa e richiede troppo tempo se il numero dei byte in memoria è grande. Si osservi che, per salvare nello stack il contenuto di due successive locazioni di memoria, sono neccessarie le istruzioni seguenti:

```
LD HL,(MEMLOC)      16 cicli T
PUSH HL             11 cicli T
POP HL              10 cicli T
LD (MEMLOC),HL      16 cicli T
                    53 cicli T = 21,2 microsecondi a 2,5 MHz
```

Un'alternativa eccellente al sistema di salvare nello stack i dati di un programma consiste nel definire un nuovo spazio dei dati per il programma che provoca l'interruzione manipolando dei puntatori in memoria, che hanno la funzione dI puntare all'inizio dello spazio riservato ai dati. Si osservi che le tre istruzioni INC IX con le quali comincia SERVN muovano verso l'alto di tre byte (in memoria) lo spazio riservato ai dati.

Dato che tutte e tre le condizioni sufficienti ad assicurare la rientranza sono pienamente soddisfatte, SERVN è un programma rientrante. Avete potuto osservare questo fatto nel corso del Passo 2 quando avete verificato che SERVN gestisce regolarmente le interruzioni nidificate.

La routine di gestione dell'interruzione, sia essa SERVN oppure una qualche altra routine, si scontra con un solo problema: Essa non dispone di alcun mezzo per evitare di essere interrotta da interruzioni dello stesso tipo di quella che gestisce. A differenza delle routine di servizio delle interruzioni mascherabili, che sono in grado di disabilitare tali interruzioni, una routine di servizio delle interruzioni non-mascherabili non è in grado di fare la stessa cosa per le interruzioni non-mascherabili.
Ne deriva che, se si presentano due interruzioni non-mascherabili in successione abbastanza rapida, la routine di servizio può iniziare l'esecuzione, essere interrotta, ricominciare l'esecuzione una seconda volta, completarla e riprendere infine la prima esecuzione dal punto nel quale l'aveva lasciata. In casi del genere è essenziale che lo spazio riservato ai dati sia protetto per evitare che si ottengano dei risultati indesiderati. Diamo infatti uno sguardo a quello che succede se non si salva lo spazio dei dati di SERVN.

Nella routine SERVN effettuate le sostituzioni che seguono:

Locazione	**Contenuto originario**	**Sostituzione**
DSN	DD	00
DSN + 1	23	00
DSN + 2	DD	00
DSN + 3	23	00
DSN + 4	DD	00
DSN + 5	23	00

Avete, in tal modo, eliminato tre istruzioni essenziali per ridefinire lo spazio dei dati a disposizione della routine SERVN, ogni volta che si entra in essa. Ne osserverete ora le conseguenze.

Passo 5

Iniziate il programma INIT1N. Generate in successione due interruzioni non mascherabili. Per quanto tempo il conteggio è mantenuto fermo questa volta?

Noi abbiamo osservato che il servizio della seconda interruzione ha richiesto il solito tempo. La gestione della prima interruzione, invece, non è stata completata in non meno di due minuti!

Passo 6

Mettiamo ora alla prova le capacità di rientro della routine di gestione delle interruzioni SERV1. Si osservi come SERV1 disabiliti le interruzioni mascherabili, una volta abbia il controllo. Per provare la sua capacità di rientro, dobbiamo variare questa caratteristica. Per darle la possibilità di salvare lo stato della CPU ed aggiornare il puntatore allo spazio dei dati, inserite l'istruzione EI alla locazione DS1+6 al posto della istruzione NOP. Il codice esadecimale associato ad EI è FB. Iniziate l'esecuzione a partire dalla locazione INIT1. Generate un certo numero di interruzioni mascherabili. Verificate la corrispondenza con la tabella seguente:

Interruzione N.	Stack pointer
0	0F00
1	0EEE
2	0EDC
3	0ECA
4	0EB8
5	0EA6

Voi dovreste osservare che le interruzioni sono nidificate e che sono state servite regolarmente.

CAPITOLO 7

IL DISPOSITIVO DI INGRESSO/USCITA PARALLELO PIO Z80

INTRODUZIONE

Il dispositivo PIO (Parallel Input/Output, Ingresso/Uscita parallelo) fa parte di una famiglia di componenti realizzati per facilitare l'intefacciamento della CPU Z80. Tale circuito programmabile consente, tramite due porte, un trasferimento di dati parallelo, compatibile TTL, tra la CPU Z80 e dei dispositivi periferici. Contenuto in un contenitore dual-in line (DIP) a 40 piedini, il componente PIO presenta le seguenti caratteristiche principali:

- Due porte indipendenti bidirezionali di interfaccia parallela con dispositivi periferici di otto bit, ciascuna con segnali di controllo di "handshake"
- I/O pilotato da interruzione
- Quattro modi di funzionamento selezionabili da software:
  Modo 0 – Uscita di un byte
  Modo 1 – Ingresso di un byte
  Modo 2 – Trasferimento bidirezionale di un byte (disponibile solo per la Porta A)
  Modo 3 – Controllo dei singoli bit
  Ad ogni modo di funzionamento corrisponde una logica di handshake con controllo da interruzione.
- Logica di interruzione prevista per una catena di priorità (daisy chain)
- Tutti gli ingressi e le uscite sono completamente compatibili TTL.

Alla fine di questo capitolo sare in grado di:

- Capire tutte le caratteristiche sopra elencate del PIO
- Verificare e provare tramite esperimenti ciascuna di tali caratteristiche
- Saper leggere il Manuale Tecnico del PIO Z80, pubblicato dalla Zilog e dalla SGS-ATES.

ASPETTI GENERALI DEL DISPOSITIVO PIO

In questa sezione vedremo in dettaglio la configurazione dei piedini del PIO, riassumeremo brevemente i suoi modi di funzionamento e descriveremo come è possibile programmarlo. Questa parte è volutamente breve, in modo da permettervi di passare velocemente agli esperimenti, in cui potrete trovare presentati e discussi i dettagli sull'uso del componente PIO.

Descrizione dei piedini del dispositivo PIO Z80 (Figura 7-1)

La Figura 7-1 presenta la configurazione dei piedini del PIO.

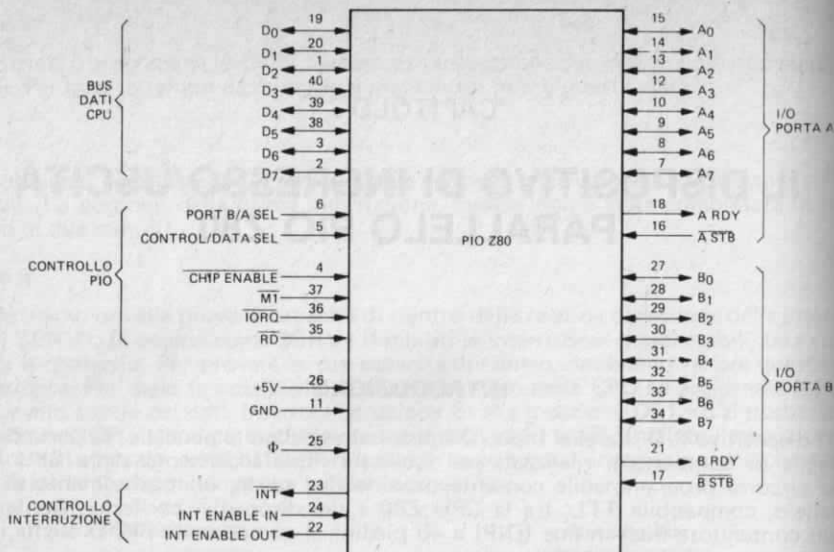


Figura 7-1. Configurazione dei piedini del dispositivo PIO.

Di seguito viene data (per gentile concessione della SGS-ATES) la descrizione delle funzioni dei singoli piedini.

D7-D0 — Bus dei dati della CPU Z80 (bidirezionale-tre stati)
Il bus viene usato per trasferire dati e comandi tra la CPU Z80 e il PIO Z80. D0 è il bit meno significativo del bus.

B/A Sel — Selezione della Porta A o B (ingresso, attivo alto).
Questo piedino definisce quale sarà la porta a cui si vuole accedere durante il trasferimento di un dato (o comando) tra la CPU Z80 e il PIO Z80. Un livello basso su questo piedino seleziona la Porta A, mentre un livello alto seleziona la Porta B. Spesso il bit A0 del bus degli indirizzi della CPU viene usato per questa funzione di selezione.

C/D Sel — Selezione di Comando o Dato (ingresso, attivo alto).
Questo piedino definisce quale tipo di dato viene trasferito fra la CPU e il PIO. Un livello alto su questo piedino durante un'operazione di scrittura da parte della CPU fa sì che il contenuto del bus dei dati dello Z80 venga interpretato come un *comando* dalla porta del PIO selezionata dalla linea di selezione B/A. Un livello basso su questo piedino sta ad indicare invece che il bus dei dati dello Z80 viene usato per il trasferimento di un *dato* fra la CPU e il PIO.
Per questa funzione viene spesso usato il bit A1 del bus degli indirizzi della CPU.

$\overline{CE}$ — Chip Enable: Abilitazione del dispositivo (ingresso, attivo basso)
Un livello basso su questo piedino consente al PIO di accettare un comando o un dato dalla CPU durante un ciclo di scrittura o di trasmettere un dato alla CPU durante un ciclo di lettura. Tale segnale è generalmente un "OR" logico dei quattro indirizzi di I/O corrispondenti alla Porta A e alla Porta B, sia comandi che dati.

Φ Clock di sistema (ingresso)
Il PIO Z80 utilizza il clock del sistema Z80 per sincronizzare internamente certi segnali. Si tratta di un clock a fase singola.

$\overline{M1}$ Machine Cycle One: Ciclo Macchina Uno, generato dalla CPU (ingresso, attivo basso)
Questo segnale, proveniente dalla CPU, viene usato come impulso di sincronizzazione per controllare numerose operazioni all'interno del PIO.
Quando $\overline{M1}$ è attivo ed è attivo anche il segnale $\overline{RD}$, la CPU Z80 è in fase di prelievo (fetch) di un'istruzione dalla memoria, mentre quando sono contemporaneamente attivi $\overline{M1}$ ed $\overline{IORQ}$, la CPU sta rispondendo ad una interruzione. Il segnale $\overline{M1}$, ha nell'ambito del PIO Z80, altre due funzioni:

1. $\overline{M1}$ sincronizza la logica di interruzione del PIO.
2. Quando è presente $\overline{M1}$ senza che né RD né $\overline{IORQ}$ siano attivi, la logica del PIO entra nello stato di reset.

$\overline{IORQ}$ Input/Output: Richiesta di Ingresso/Uscita, generato dalla CPU Z80 (ingresso, attivo basso)
Il segnale $\overline{IORQ}$ viene usato unitamente alla linea di selezione B/A, alla linea di selezione C/D, ed ai segnali $\overline{CE}$ e $\overline{RD}$ per il trasferimento di dati e comandi tra la CPU Z80 e il dispositivo PIO Z80. Quando $\overline{CE}$, $\overline{RD}$ e $\overline{IORQ}$ sono attivi, la porta indirizzata dalla linea B/A trasferisce un dato alla CPU (si tratta di un'operazione di lettura). Viceversa, quando $\overline{CE}$ e $\overline{IORQ}$ sono attivi, ma $\overline{RD}$ non lo è, la porta indirizzata dalla linea B/A riceverà dalla CPU un dato o un comando, secondo quanto indicato dal segnale di selezione C/D. Ancora, se $\overline{IORQ}$ e $\overline{M1}$ sono contemporaneamente attivi, la CPU sta rispondendo ad una interruzione (Interrupt Acknowledge, riconoscimento dell'interruzione) e la porta che ha generato l'interruzione porrà automaticamente il suo vettore di interruzione sul bus dei dati della CPU, posto che tale porta sia il dispositivo di priorità più elevata tra quelli che abbiano richiesto un'interruzione.

$\overline{RD}$ Read: Ciclo di Lettura (ingresso, attivo basso).
Quando $\overline{RD}$ è attivo significa che è in corso un'operazione di lettura in memoria (Memory Read) o di lettura in un dispositivo di ingresso (I/O Rread). $\overline{RD}$ viene usato, con i segnali di selezione B/A e C/D e con i segnali $\overline{CE}$ e $\overline{IORQ}$, per trasferire i dati dal PIO Z80 alla CPU Z80.

IEI Interrupt Enable In: Ingresso di Abilitazione dell'Interruzione (ingresso, attivo alto)
Questo segnale viene impiegato per formare una catena di priorità delle interruzioni (del tipo daisy chain) quando è presente più di un dispositivo in grado di generare interruzione. Il livello alto su questo piedino indica che non è in corso di esecuzione nessuna routine di risposta all'interruzione per altri dispositivi di priorità più elevata.

IEO Interrupt Enable Out: Uscita di Abilitazione dell'Interruzione (uscita, attivo alto).
Il segnale IEO è l'altro segnale necessario per formare una catena di priorità daisy chain. E' alto solo se il segnale IEI è alto e la CPU non sta servendo una interruzione di questo PIO, blocca quindi i dispositivi a priorità più bassa, impedendo così che generino una interruzione durante l'esecuzione di una routine di risposta all'interruzione relativa ad un dispositivo a più alta priorità.

$\overline{INT}$ Interrupt Request: Richiesta di interruzione (uscita, open drain, attivo basso).

Quando è attivo, il PIO Z80 sta richiedendo una interruzione alla CPU Z80.

A0-A7 — Bus della Porta A (bidirezionale, tre stati).
Questo bus a 8 bit viene usato per il trasferimento di dati e/o di informazioni di stato o di comando tra la Porta A del PIO Z80 e un dispositivo periferico. A0 è il bit meno significativo del bus dei dati della Porta A.

$\overline{\text{A STB}}$ — Impulso di Strobe (sincronizzazione) della Porta A, generato dal dispositivo periferico (ingresso, attivo basso).
La funzione di questo segnale dipende dal modo di funzionamento selezionato per la Porta A:

1. Modo d'uscita: Il fronte positivo di questo segnale di strobe viene fornito dalla periferica per comunicare che ha ricevuto e caricato il dato reso disponibile in uscita dal PIO.
2. Modo d'ingresso: Il segnale di strobe è fornito dalla periferica per caricare il dato nel registro di ingresso della Porta A. Il dato è caricato all'interno del PIO quando il segnale è attivo.
3. Modo bidirezionale: Quando questo segnale è attivo, il dato viene caricato dal registro di uscita della Porta A sul bus bidirezionale di quest'ultima. Il fronte positivo del segnale di strobe comunica l'avvenuta ricezione del dato.
4. Modo di controllo: Il segnale di strobe è disabilitato internamente.

A RDY — Register A Ready: Registro A Pronto (uscita, attivo alto).
Il significato e la funzione di questo segnale dipendono dal modo di funzionamento selezionato per la Porta A:

1) Modo di uscita: Questo segnale diventa attivo per indicare che il registro di uscita della Porta A è stato caricato e il bus dei dati in uscita è stabile e pronto per il trasferimento al dispositivo periferico.
2) Modo di ingresso: Il segnale è attivo quando il registro di ingresso della Porta A è vuoto ed è pronto a ricevere i dati provenienti dal dispositivo periferico.
3) Modo bidirezionale: Questo segnale è attivo quando nel registro di uscita della Porta A i dati sono disponibili per essere inviati al dispositivo periferico. In questo modo di funzionamento, i dati non vengono caricati nel bus dei dati della Porta A, finchè $\overline{\text{A STB}}$ non sia attivo.
4) Modo di controllo: Il segnale è disattivato e mantenuto allo stato basso.

B0-B7 — Bus della Porta B (bidirezionale, tre stati). Questo bus a 8 bit viene utilizzato per trasferire i dati e/o le informazioni di stato o di comando tra la Porta B del PIO e il dispositivo periferico. Il bus dati della Porta B è in grado di fornire 1,5 mA a 1,5 V per pilotare transistor Darlington. B0 è il bit meno significativo del bus.

$\overline{\text{B STB}}$ — Impulso di Strobe della Porta B, generato dal dispositivo periferico (ingresso, attivo basso).
La funzione di questo segnale è simile a quella del segnale A STB con la seguente eccezione: *nel modo di funzionamento bidirezionale della Porta A,* questo segnale effettua il caricamento dei dati provenienti dal dispositivo periferico nel registro di ingresso della Porta A.

B RDY — Registro B Ready: Registro B Pronto (uscita, attivo alto).
La funzione di questo segnale è simile a quella del segnale A Ready con la seguente eccezione: *nel modo di funzionamento bidirezionale*

della Porta A, questo segnale è alto quando il registro di ingresso della Porta A è vuoto e pronto a ricevere i dati provenienti dal dispositivo periferico.

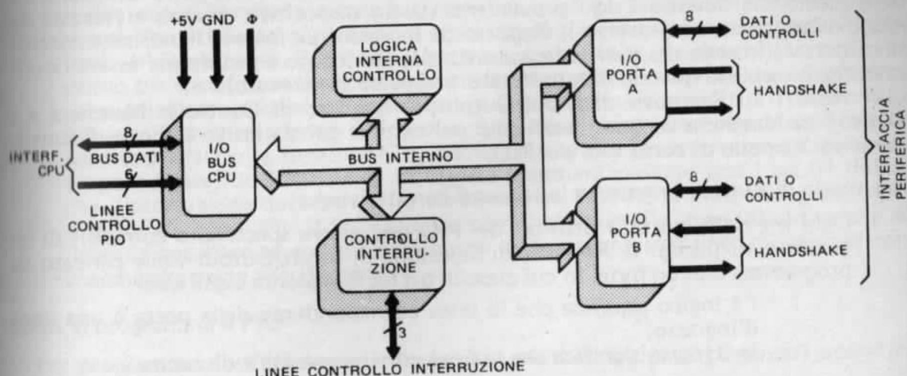


Figura 7-2. Schema a blocchi funzionale del PIO.

Descrizione funzionale del PIO

La Figura 7-2 contiene uno schema a blocchi funzionale del dispositivo PIO. L'interfaccia con la CPU è formata dalle 8 linee del bus dei dati e dalle seguenti linee di controllo: $\overline{M1}$, $\overline{IORQ}$, $\overline{RD}$, B/A Sel, C/D Sel e $\overline{CE}$. Il bus dei dati è utilizzato per trasferire tutti i byte di dati e di comandi tra la CPU e il PIO, mentre le linee di controllo abilitano il PIO ($\overline{CE}$), definiscono la natura del byte da trasferire (C/D), selezionano la Porta (B/A) e specificano la direzione del flusso dei dati ($\overline{M1}$, $\overline{IORQ}$ e $\overline{RD}$). All'interno del PIO, la parte di interfaccia con la CPU comunica con gli altri blocchi funzionali mediante il bus dei dati interno del PIO. La parte di controllo delle interruzioni gestisce le tre linee di controllo dell'interruzione: IEI, IEO e $\overline{INT}$, mentre la Logica Interna di Controllo svolge la necessaria funzione di sincronizzazione e di coordinamento dell'insieme.
Le funzioni di I/O delle Porte A e B consentono l'interfacciamento del PIO con i dispositivi periferici. Le otto linee dati o comandi che compongono il bus (periferico) delle porte di I/O, sono:

Per la Porta A: PA0, PA1, PA2, PA3, PA4, PA5, PA6 e PA7
Per la Porta B: PB0, PB1, PB2, PB3, PB4, PB5, PB6 e PB7

Le linee di "handshake" sono invece le seguenti:

Per la Porta A: $\overline{ASTB}$ e ARDY
Per la Porta B: $\overline{BSTB}$ e BRDY

La funzione di I/O di ciascuna delle due porte può essere suddivisa in un ulteriore schema a blocchi funzionale (Figura 7-3).

La descrizione che segue, essendo le Porte A e B del PIO di fatto identiche (ad eccezione del modo di funzionamento bidirezionale che può essere usato con la sola Porta A) farà riferimento ad una sola porta senza distinguere tra la Porta A e la Porta B. Quanto detto potrà quindi essere riferito, indistintamente, a ciascuna delle due porte presenti nel dispositivo PIO.

Il Registro di Controllo Modo contiene il codice a due bit che specifica il modo di funzionamento della porta del PIO e viene caricato da programma, attraverso un byte di comando inviato dalla CPU al PIO. Nel prossimo paragrafo vedremo come procedere per specificare tale modo di funzionamento. Tutti i dati trasmessi dal PIO ad un dispositivo esterno devono passare attraverso il Registro di Uscita dei Dati a otto bit, mentre i dati provenienti da un dispositivo esterno e ricevuti dal PIO devono passare attraverso il Registro di Ingresso. Le linee di handshake controllano il trasferimento dei dati indicando sia quando il dato è pronto per essere ricevuto (o trasmesso) sia quando il dato è stato trasmesso (o ricevuto).

Il registro di Selezione di Input/Output, il Registro di Controllo Maschera e il Registro di Maschera vengono usati solo nel caso in cui sia attivo il modo di funzionamento 3 (modo di controllo) del PIO.

Tale modo di funzionamento ha le seguenti caratteristiche:

1. Ogni bit della porta a otto bit del PIO può essere specificato come bit di ingresso o di uscita. Il Registro di Selezione di Input/Output viene caricato da programma con un byte, in cui ciascun bit ha il seguente significato:

    l'1 logico significa che la linea corrispondente della porta è una linea d'ingresso;
    lo 0 logico significa che la linea corrispondente è di uscita.

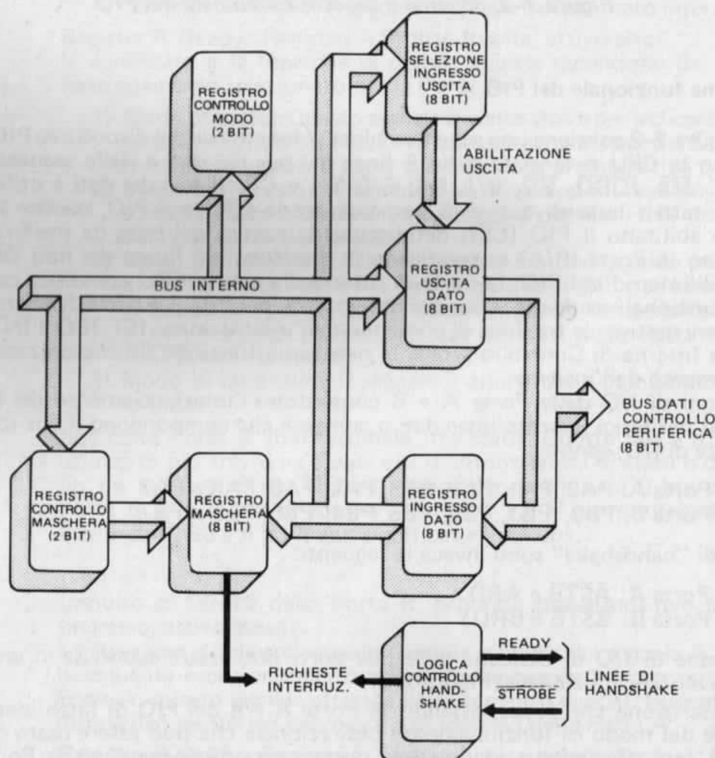


Figura 7-3. Schema a blocchi funzionale di una parte di I/O del PIO.

2. Nel modo di funzionamento 3, il PIO controlla istante per istante se sulle linee della porta è presente una particolare configurazione. In caso affermativo il PIO genera automaticamente una richiesta di interruzione alla CPU Z80. La configurazione "di confronto" può essere definita dal programma utilizzando il Registro di Maschera e il Registro di Controllo Maschera presenti nel PIO. Il Registro di Maschera, a otto bit, specifica quali dei piedini della porta devono essere "tenuti sotto controllo" e quali devono invece essere mascherati (ignorati). Il Registro di Controllo Maschera è costituito da due bit soltanto. Il primo bit specifica se deve essere generata interruzione quando i segnali presenti sui piedini non maschèrati sono posti a 0 o quando sono posti a 1. Il secondo bit specifica se alle linee di ingresso non mascherate deve essere applicata la funzione AND o la funzione OR, come criterio per generare una interruzione. In particolare, un Registro di Controllo Maschera caricato con i bit 01 indica che l'interruzione deve essere generato solo se TUTTE le linee di ingresso non mascherate sono allo 0 logico, così come, al contrario, uno 00 indica che la interruzione deve essere generata anche se esiste una sola linea di ingresso non mascherata posta allo 0 logico.

Come si programma il PIO

Per programmare il PIO occorre distinguere tre diversi tipi di parametri operativi del PIO:

1. *Il Vettore di Interruzione:* il dispositivo PIO può generare le interruzioni secondo il Modo 2 della CPU Z80.
   Ogni volta che interrompe la CPU, il PIO deve così fornire un codice di identificazione (id) del dispositivo. (Per una descrizione più dettagliata della gestione delle interruzioni nel Modo 2 dello Z80, si rimanda al Capitolo 6).
   Il valore id viene interpretato dalla CPU come il byte meno significativo di un indirizzo alla tabella di vettori, ciascuno dei quali a sua volta, punta all'inizio di una routine di gestione dell'interruzione. Il codice id viene caricato nel PIO scrivendolo in un registro associato alla porta che si vuole selezionare. Il bit meno significativo del codice id deve essere zero, in modo da indicare al PIO che il byte di controllo trasferito dalla CPU è un vettore di interruzione.
2. *Il Modo di Funzionamento:* le porte del PIO possono operare in uno dei quattro modi seguenti: Modo 0 (uscita di un byte), Modo 1 (ingresso di un byte), Modo 2 (trasferimento bidirezionale) o Modo 3 (controllo).
   Per selezionare il modo di funzionamento è necessario inviare, come byte di controllo alla porta desiderata, un byte dal seguente formato:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

   Dove M1 e M0 hanno il significato seguente:
   00 indica il modo uscita (Modo 0)
   01 indica il modo ingresso (Modo 1)
   10 indica il modo bidirezionale (Modo 2)
   11 indica il modo di controllo (Modo 3)

   Il fatto che tutti i bit di ordine inferiore (D0-D3) siano posti ad uno serve od indicare al PIO che il byte di controllo inviato specifica il modo di funzionamento e l'informazione è quindi diretta al Registro di Controllo Modo.
   Se per una data porta viene selezionato il Modo 3, il successivo byte di controllo per la stessa porta viene interpretato dalla logica di controllo del PIO come byte di Selezione di Input/Output (e caricato quindi nel registro corrispondente) e definirà così quali sono le linee di ingresso e quali quelle di uscita.

3. *Parola di Controllo delle Interruzioni:* per ciascuna delle due porte la parola di controllo delle interruzioni (un byte) ha il seguente formato:

D7	D6	D5	D4	D3	D2	D1	D0
Abilita le interruzioni	AND/OR	Alto/ basso	Segue maschera	0	1	1	1
	usati solo nel Modo 3						

Vediamo ora meglio il significato di ciascuno di questi bit:

D7 — Abilita le interruzioni (Enable Interrupt): Questo bit definisce lo stato di un flip-flop di abilitazione delle interruzioni della porta del PIO. Se esso è posto ad uno, la porta del PIO sarà abilitata a generare interruzioni, se invece è 0, tutte le richieste di interruzioni provenienti dal dispositivo periferico verranno ignorate. Va notato che questo bit è completamente indipendente dai flip-flop IFF1 e IFF2 presenti all'interno della CPU Z80.

D6 — AND/OR: Questo bit viene usato solo nel Modo 3 e specifica il criterio di generazione dell'interruzione. Se è posto a 1 implica che *tutte* le linee non mascherate della porta devono essere attive perchè l'interruzione possa essere generata: è cioè applicata una funzione AND. Se è posto a 0, invece, indica che è sufficiente l'esistenza di un bit attivo non mascherato per generare una interruzione; è il caso cioè di una funzione OR.

D5 — Alto/basso: Questo bit viene usato solo nel Modo 3 e indica quale stato logico deve essere considerato attivo (0 o 1) per le linee della porta.

D4 — Segue maschera: Se questo bit è a 1 ed è selezionato il Modo 3, il successivo byte di controllo dovrà essere interpretato dal PIO come byte di maschera. Le linee della porta, cui corrispondono nel byte di maschera bit posti a 1, vengono ignorate, mentre quelle il cui bit sia a 0 vengono considerate attive e "tenute sotto controllo" ai fini della generazione dell'interruzione (secondo le modalità descritte poco sopra a proposito dei bit D7, D6 e D5).
Se il modo di funzionamento non è il Modo 3, il bit D4 posto ad 1 farà sì che le interruzioni precedenti (in attesa cioè di essere servite) vengano cancellate (reset).

D3-D0: Quando questi bit sono posti a 0111, il PIO interpreta il byte di controllo come la Parola di Controllo delle Interruzioni.

Negli esperimenti che vedremo in seguito, potrete esercitarvi in quasi tutte le caratteristiche finora illustrate del dispositivo PIO; preferiamo perciò rimandare alle pagine successive una discussione più approfondita.

Inizializzazione del dispositivo PIO (Reset)

Come la CPU Z80, anche il dispositivo PIO deve entrare nello stato di reset al momento dell'accensione. L'invio del segnale $\overline{M1}$ basso senza che siano attivi nè $\overline{RD}$ nè $\overline{IORQ}$ fa sì che il PIO entri in uno stato di reset non appena il segnale $\overline{M1}$ diventa inattivo (ritorna ad 1). Lo stato di reset viene definito nel modo seguente:

1. Entrambi i registri di maschera della porta vengono posti a zero.
2. Tutte le linee dei dati della porta vengono poste in uno stato di alta impedenza.
3. Le linee di "handshake" RDY vengono disattivate (basse)
4. Viene selezionato il Modo 1.
5. Vengono posti a zero i flip-flop di abilitazione dell'interruzione della porta.
6. I registri di uscita della porta vengono posti a zero.
7. I registri di indirizzo del vettore dell'interruzione non vengono modificati (e all'accensione potranno contenere perciò dei dati casuali).

Lo stato di reset viene mantenuto finchè il PIO non riceve dalla CPU una parola di controllo.
Tutti i dati fin qui forniti sono contenuti anche nel *Manuale Tecnico del PIO Z80* pubblicato dalla Zilog e dalla SGS-ATES. Tale manuale è ben scritto, molto bene organizzato, e contiene un concisa ma esauriente trattazione di tutte le caratteristiche funzionali e operative del chip PIO: vi consigliamo di acquistarlo se intendete approfondire il lavoro con questo componente.
Gli esperimenti che seguono sono stati studiati per integrare quanto illustrato in tale manuale: vi sono presenti molti esempi di circuiti che fanno uso del PIO e che non sono presi in esame nella documentazione fornita dai costruttori.

INTRODUZIONE AGLI ESPERIMENTI

La scheda CPU del Nanocomputer contiene due componenti PIO (Figura 7-4). Il PIO N. 1 viene usato per interfacciare la CPU al miniterminale tastiera/display del Nanocomputer e alle porte seriali dell'unità a cassetta e della telescrivente (per maggiori dettagli si rimanda al Capitolo 5). Il PIO N. 2, invece, non è usato dal

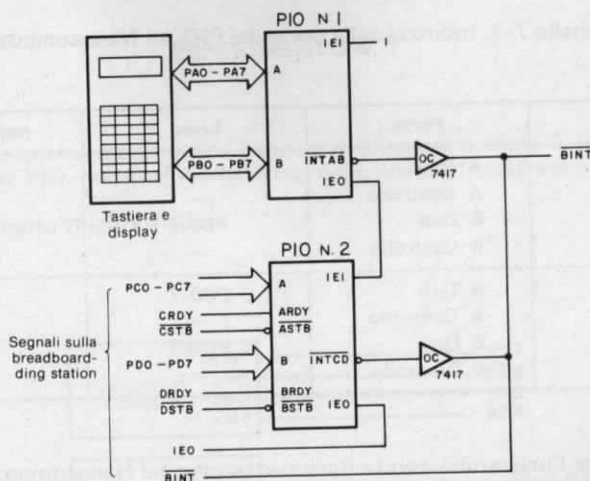


Figura 7-4. Disposizione dei PIO sul Nanocomputer.

programma Monitor del Nanocomputer ed è disponibile per essere utilizzato in circuiti esterni. E' accessibile all'utente mediante un cavo che, partendo dal connettore J7 posto sulla scheda del Nanocomputer, si collega al connettore PIO presente sulla scheda per esperimenti (a fianco del connettore J2). Sullo zoccolo C, a quaranta piedini, sono disponibili poi i seguenti segnali:

Porta A del PIO N. 2 (parte dati): PC0, PC1, PC2, PC3, PC4, PC5, PC6 e PC7
Porta A del PIO N. 2 (parte handshake): $\overline{\text{CSTB}}$, CRDY
Porta B del PIO N. 2 (parte dati): PD0, PD1, PD2, PD3, PD4, PD5, PD6 e PD7
Porta B del PIO N. 2 (parte handshake): $\overline{\text{DSTB}}$, DRDY.

Va notato che le linee della Porta A del PIO N. 2 sono etichettate con la lettera "C" mentre le linee della Porta B del PIO N. 2 sono etichettate con la lettera "D", questo per distinguere i segnali del PIO N. 2 da quelli del PIO N. 1.

Ne consegue che, negli schemi elettrici e negli esperimenti che seguono, PAn, PBn, $\overline{ASTB}$, ARDY, $\overline{BSTB}$ e BRDY (dove n = 0, . . ., 7) saranno etichettati rispettivamente Pcn, PDn, $\overline{CSTB}$, CRDY, $\overline{DSTB}$ e DRDY.
Gli altri piedini del PIO sono collegati invece alla scheda del Nanocomputer. Gli schemi nell'Appendice mostrano come sono realizzate queste connessioni.
Il piedino $\overline{CE}$ è collegato alla linea $\overline{IOQ2}$, mentre i piedini B/A e C/D sono collegati alle linee di indirizzo BA0 e BA1.
Ne risulta che il PIO N. 2 viene selezionato ogniqualvolta le linee di indirizzo, da BA7 a BA2, sono uguali a:

BA7	6	5	4	3	BA2
0	0	0	0	1	0

cioè, il valore che attiva $\overline{IOQ2}$. Le due linee di indirizzo BA0 e BA1 stabiliscono se la porta indirizzata del PIO N. 2 è la Porta A o la Porta B, e se il byte trasferito deve essere considerato un comando o un dato. L'indirizzamento del PIO N. 1 e del PIO N. 2 può essere sintetizzato nel modo seguente (Tabella 7-1):

Tabella 7-1. Indirizzi delle porte dei PIO sul Nanocomputer.

PIO N.	Porta	Linee	Indirizzo (esa)
1	A Dati	PA0–7	04
	A Controllo	–	06
	B Dati	PB0–7	05
	B Controllo	–	07
2	A Dati	PC0–7	08
	A Controllo	–	0A
	B Dati	PD0–7	09
	B Controllo	–	0B

Per mantenere l'uniformità con la documentazione del Nanocomputer, la Porta A del PIO N. 2 verrà chiamata Porta C e la Porta B dello stesso verrà chiamata Porta D.
Per quanto riguarda la CPU Z80 le linee del bus dei dati vengono collegate direttamente al bus dei dati del PIO e le linee di controllo del PIO, $\overline{M1}$, $\overline{IORQ}$ e $\overline{RD}$ sono anch'esse collegate direttamente (in realtà tramite dei componenti di buffer) ai piedini dello Z80 indicati con lo stesso simbolo. Il clock Φ dello Z80 è collegato all'ingresso di clock del PIO (piedino 25).
Le linee di controllo dell'interruzione, IEI, IEO e $\overline{INT}$ verranno esaminate in seguito. E' sufficiente per ora dire che vengono usate all'interno di una catena di priorità formata con il PIO N. 1, comportando così una struttura di priorità nella risposta alle interruzioni provenienti da dispositivi periferici. I restanti peidini del PIO N. 2 rappresentano i bus dei dati della porta e i segnali di handshake che vengono inviati alla scheda per esperimenti del Nanocomputer.

Nota: In questi esperimenti, allo scopo di uniformarci con la documentazione del Nanocomputer, sono state usate le designazioni C e D per le porte del PIO. E' estremamente importante che sappiate usare sia le notazioni utilizzate dai fabbricanti dei dispositivi sia quelle del Nanocomputer.

Gli esperimenti che eseguiremo possono essere così presentati:

Esperimento N.	Commenti
1	Illustra il Modo 0 di funzionamento del PIO senza segnali di handshake
2	Illustra il Modo 0 di funzionamento del PIO con segnali di handshake
3	Esamina ulteriormente la funzione dei segnali di handshake nel Modo di funzionamento 0
4	Illustra il Modo 1 di funzionamento del PIO
5	Illustra il Modo 2 di funzionamento del PIO
6	Illustra il Modo 3 di funzionamento del PIO
7	Illustra la priorità esistente tra due porte dello stesso PIO
8	Illustra come collegare in daisy chain il PIO

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è quello di illustrare il Modo 0 di funzionamento del componente PIO, non facendo però uso delle linee di handshake.

Schema del circuito (Figura 7-5)

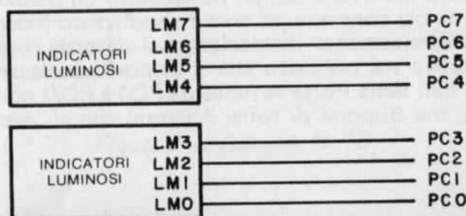


Figura 7-5. Schema N. 1A.

Programma OUTSIM

Codice oggetto		Codice sorgente	Commenti
3E0F	OUTSIM:	LD A,0FH	; Programma in Modo 0 il PIO N. 2
D30A		OUT (0AH),A	
3E43		LD A,43H	; Poni in uscita sulle linee PC0-7 il byte 43H
D308		OUT (08H),A	
76		HALT	

Passo 1

Caricate la routine OUTSIM. La prima istruzione di uscita OUT mette in uscita il byte 0F sulla Porta 0A, cioè la parte di controllo della Porta A del PIO N. 2 (o Porta C del PIO).
La presenza di un 1 logico in corrispondenza dei quattro bit meno significativi del byte (0F) comunica al PIO che 0F è un byte di comando per l'impostazione del modo di funzionamento della porta del PIO. Il modo viene specificato dai due bit più significativi. Poichè i bit D7 e D6 sono entrambi zero, il modo di funzionamento richiesto è il Modo 0, o Modo di Uscita.
Ricordate che la parola di controllo per l'impostazione del modo di funzionamento è:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

Il secondo byte che il programma OUTSIM pone in uscita (seconda istruzione OUT) 43 ed è inviato alla porta 08, cioè alla Porta A (dati) del PIO N. 2 (o porta C del PIO). Come risultato la Porta C del PIO memorizzerà il dato in modo del tutto analogo ad una comune porta costituita da latch. Il codice 43 sarà quindi visualizzato sugli indicatori luminosi alla fine della esecuzione del programma.

Passo 2

Eseguite il programma sopra indicato iniziando dalla locazione OUTSIM. Cosa notate?

Nel nostro caso la visualizzazione sugli indicatori luminosi LM0-LM7 indicava il codice 43. Va osservato che il programma ha eseguito solo una istruzione di uscita dati in direzione della Porta C del PIO e quindi ha eseguito un'istruzione di HALT. Poichè sugli indicatori luminosi continua ad essere visualizzato il dato in uscita. è dimostrato che il PIO ha effettivamente memorizzato il dato da trasferire all'esterno. Il significato di questa prova sta nel fatto che il dispositivo esterno collegato al chip PIO tramite il bus dei dati della Porta A (linee da PC0 a PC7) non deve quindi catturare "al volo" il dato, ma dispone di tutto il tempo che gli necessita per leggere il dato.

Passo 3

Usando un filo attivate $\overline{M1}$ mettendo in contatto per un attimo $\overline{BM1}$ con GND. Sostituite quindi le due istruzioni alle locazioni OUTSIM e OUTSIM+1 con NOP. Con ciò avete tolto le istruzioni che programmano il PIO. Eseguite infine il programma partendo ancora da OUTSIM. Cosa notate ?

La prima cosa che abbiamo notato è stata la mancata visualizzazione del codice 43 sugli indicatori luminosi. La ragione sta nel fatto che il PIO seleziona automaticamente il Modo 1 di funzionamento quando si verifica un reset. Il livello basso di $\overline{M1}$

fa sì che il PIO venga resettato, così come accade al momento dell'accensione. Ne consegue che sarà necessario programmare di nuovo il PIO dopo ogni reset a meno che non si desideri che il dispositivo funzioni nel Modo 1.
Nel nostro esperimento abbiamo notato che, senza usare nessuna linea di handshake, il PIO può svolgere la funzione di un latch di uscita senza praticamente bisogno di componenti logici esterni.

ESPERIMENTO N. 2

Scopo

Scopo di questo esperimento è di illustrare l'uso delle linee di handshake nel Modo 0 di funzionamento del PIO.

Schema del circuito (Figura 7-5 e 7-6)

Aggiungere allo schema i due indicatori luminosi ed il collegamento tra P0 e $\overline{CSTB}$ illustrato nello schema 1B.

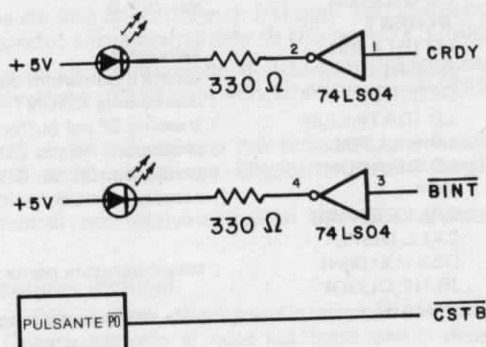


Figura 7-6. Schema N. 1B.

Programmi INITOC, MAIN e SERVOC

Codice oggetto	Codice sorgente		Commenti
ED5E	INITOC:	IM2	; predispone il modo di interruzione dello ; Z80
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; predispone il registro di interruzione
FD21E803		LD IY,SERVOC	; indirizzo routine di servizio per l'uscita ; dati dal PIO
FD22060F		LD (TABLE+06H),IY	; trasferiscilo nella tabella dei vettori
3E06		LD A,06H	; carica il vettore di interruzione

D30A		OUT (0AH),A	; relativo alla porta C
08		EX AF,AF'	; predisposizione del formato per CONVDI
3E40		LD A,40H	
08		EX AF,AF'	
3E0F		LD A,0FH	; predisposizione del modo del PIO
D30A		OUT (0AH),A	
3E87	ENPIO:	LD A87H	; abilita le interruzioni del PIO
D30A		OUT (0AH),A	
3EFF		LD A,0FFH	; inizializza il segnale CRDY
D308	THROW:	OUT (08H),A	
C3C302		JP MAIN	; salta alla routine MAIN

Codice oggetto		Codice sorgente	Commenti
FB	MAIN	EI	; abilita le interruzioni
DD21000C		LD IX,DSTACK	; fondo dello stack dei dati
DD3600FF		LD (IX+00H),0FFH	; temporizzatore per la visualizzazione
21E50F		LD HL,ADDH	; predispone il puntatore del buffer
ED57		LD A,I	; trova il valore di IFF2
EAD802		JP PE,HIGH	
3600	LOW:	LD (HL),00H	; valore = 0
1802		JR NEXT	
3610	HIGH:	LD (HL),10H	; valore = 1
2B	NEXT:	DEC HL	; sposta il puntatore del buffer
35		DEC (HL)	; decrementa COUNT
ED73E20F		LD (DATAL),SP	; trascrivi SP nel buffer
21B90F		LD HL,LEDL	; predisponi HL per CONVDI
11E50F		LD DE,ADDH	; predisponi DE per CONVDI
00	DISAB:	NOP	; nessuna operazione
CD7CFA		CALL CONVDI	
CD09F9	DLOOP:	CALL DISPL	
DD3500		DEC (IX+00H)	; temporizzatore per la visualizzazione
20F8		JR NZ,DLOOP	
C3C302		JP MAIN	; salta all'inizio del programma

Codice oggetto		Codice sorgente	Commenti
E5	SERVOC:	PUSH HL	; salva lo stato dei registri della CPU
F5		PUSH AF	
3AE40F		LD A,(ADDL)	; carica in A il valore del buffer
D308		OUT (08H),A	; poni in uscita il valore del buffer
F1		POP AF	; ripristina lo stato dei registri
E1		POP HL	; della CPU
FB		EI	; abilita le interruzioni
ED4D		RETI	; ritorno dall'interruzione

Passo 1

Collegate il circuito mostrato. I segnali di handshake ARDY e $\overline{ASTB}$ sono stati studiati per permettere al sistema (PIO e CPU) di effettuare operazioni di I/O controllate da interruzione. Con questi termini si vuole indicare un particolare metodo di controllo del flusso dei dati tra la CPU e il dispositivo esterno. Poichè è necessario questo

controllo del flusso dei dati? Vediamo, a questo proposito, il seguente programma illustrativo, che effettua l'operazione di inviare 256 byte di dati ad un dispositivo esterno:

```
DUMP:   LD    A,0FH     ; Programma il PIO per il modo di uscita
        OUT   (0AH),A   ; Invia il byte di selezione del modo di uscita alla Porta A
                        ; (parte di controllo)
        LD    HL,0F800H ; HL = indirizzo d'inizio del blocco di dati da inviare in
                        ; uscita
        LD    B,0FFH    ; B = contatore dei byte
LOOP:   LD    A,(HL)    ; Poni in uscita i byte uno alla volta
        OUT   (08H),A
        INC   HL
        DJNZ  LOOP
        HALT
```

Presupponendo che la CPU Z80 funzioni a 2,5 MHz, una volta che il PIO è stato programmato, il programma DUMP completa la trasmissione dei 256 byte in circa 3,29 millisecondi.
I dispositivi esterni come le unità a nastro, le stampanti, le telescriventi (TTY) e i video-terminali (CRT) funzionano a velocità ben diverse. Solo qualche esempio:

TTY – effettua da una operazione di I/O ogni 100 millisecondi (10 cps, caratteri per secondo) a una operazione di I/O ogni 33,3 millisecondi (30 cps)

CRT – effettua da una operazione di I/O ogni 33,3 millisecondi (300 baud) a una operazione di I/O ogni 408,16 millisecondi (19200 baud).

Si comprende quindi come il problema più serio per le operazioni di I/O sia soprattutto un problema di temporizzazione. Dispositivi esterni lenti devono poter comunicare con CPU molto veloci.
Esistono diversi metodi per risolvere questo tipo di problemi. In questo testo ne vedremo due:

Metodo 1: Interrogazione (Polling)

La CPU esegue un'operazione di polling (interrogazione, consultazione) sul dispositivo esterno. Questo metodo si basa sul fatto che il programma in esecuzione sulla CPU continui ad interrogare il dispositivo esterno per sapere se questo è pronto per ricevere o trasmettere un altro byte. A causa della differenza di velocità, la CPU dovrà interrogare più volte e riceverà più risposte negative prima che il dispositivo esterno emetta un segnale di "pronto". Solo quando il dispositivo esterno sarà pronto, la CPU potrà porre in uscita (o leggere) un nuovo byte. Va notato che il segnale di pronto (Ready) è di solito un bit, o un flag esterno, usato per indicare lo stato di "pronto" e "non pronto" della periferica e accessibile alla lettura da parte della CPU.

Metodo 2: Interruzione

Questo metodo si basa sul fatto che il dispositivo esterno non appena è pronto a ricevere (o trasmettere) il byte successivo faccia una richiesta di interruzione alla CPU. Ricevuta la richiesta la CPU pone in uscita il byte. La differenza tra questo metodo e quello precedente consiste nel fatto che, dopo aver inviato (o ricevuto) il byte di dati, la CPU potrà eseguire altre elaborazioni (ad esempio calcolare il byte successivo da porre in uscita) invece di eseguire l'operazione di polling sul dispositivo esterno attendendo che questo diventi ancora disponibile.

E' ovvio che il polling risolve il problema della differenza di velocità rallentando la CPU, con la conseguenza di ridurre la capacità di elaborazione della CPU stessa.

Se il dispositivo esterno fosse una telescrivente funzionante ad una velocità di 100 baud, l'85% delle risorse della CPU, cioè del suo tempo di elaborazione, verrebbe sprecato. Non si tratta evidentemente di un fatto positivo, ma i costi sempre più contenuti della CPU e delle loro capacità elaborative rendono il fenomeno comunque meno grave di quanto possa sembrare a prima vista.

Passo 2

Per questo esperimento useremo il PIO N. 2 in modo da effettuare operazioni di I/O controllate in interruzione dalla CPU Z80. La routine MAIN rappresenta le operazioni di elaborazione che vengono eseguite dallo Z80 tra i trasferimenti di dati. Voi stessi sarete il dispositivo esterno. Gli scambi di informazioni fra voi e la CPU Z80 procederanno nel modo seguente:

1. La CPU è occupata in elaborazioni.
2. Il dispositivo esterno (voi) richiede un dato dalla CPU generando una interruzione.
3. La CPU sospende temporaneamente l'elaborazione in corso e trasferisce il controllo ad una routine di servizio della interruzione.
4. La routine di servizio della interruzione pone in uscita un byte di dati verso il dispositivo esterno.
5. La CPU riprende l'elaborazione.

La Figura 7-7 illustra la sequenza di eventi, fin qui descritta, per operazioni di I/O controllate da interruzione.

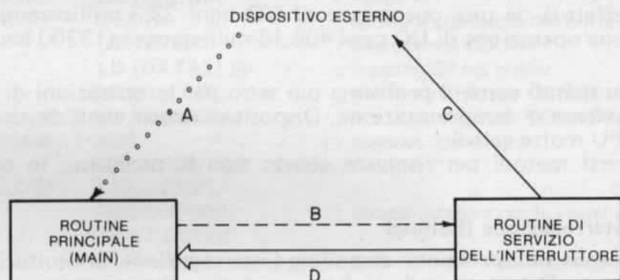


Figura 7-7. Uscita di dati controllata in interruzione. Il segnale di interruzione è indicato con A. La linea B rappresenta il trasferimento del controllo che si verifica come risposta all'interruzione. Il dato posto in uscita dalla routine di servizio dell'interruzione è indicato con C, mentre D indica il ritorno del controllo al programma principale (MAIN).

Vediamo ora più dettagliatamente, la situazione rappresentata in Figura 7-7, individuando il ruolo particolare rivestito dal dispositivo PIO e la funzione dei segnali di handshak ($\overline{\text{ASTB}}$ ($\overline{\text{CSTB}}$) e ARDY (CRDY). Notate che quella che daremo non è che una delle molte descrizioni possibili. Si veda la NOTA alla fine del PASSO 3 di questo esperimento:

1. La CPU sta eseguendo il programma MAIN.
2. Il dispositivo esterno richiede un'uscita di dati dal PIO portando bassa la linea $\overline{\text{CTSB}}$.
3. Il PIO, avendo riconosciuto il segnale $\overline{\text{CSTB}}$ attivo, attiva la linea di interruzione della CPU Z80, richiedendo così una interruzione. Contemporaneamente, il PIO porta basso (non attivo) il segnale CRDY, per fare in modo che il dispositivo esterno non legga il byte in uscita prima che tale dato non sia pronto.

4. La CPU Z80 riceve la richiesta di interruzione e invia al PIO un segnale di riconoscimento (acknowledge) dell'interruzione, attivando contemporaneamente $\overline{M1}$ e $\overline{IORQ}$.
5. Il PIO pone sul bus dei dati D0–D7 il codice di identificazione, cioè il vettore d'interruzione.
6. La CPU, che è programmata per servire le interruzioni nel Modo 2, legge il codice di identificazione del dispositivo e, interpretandolo come il byte meno significativo di un indirizzo della tabella dei vettori che puntano alla routine di servizio delle interruzioni, trasferisce il controllo alla routine di servizio della interruzione SERVOC.
7. La routine SERVOC pone in uscita un byte sulla Porta A (dati) del PIO N. 2, linee PC0–PC7.
8. Il PIO comunica al dispositivo esterno la disponibilità del byte in uscita ponendo il pin CRDY ad 1.
9. Il dispositivo esterno può leggere a questo punto il byte in qualunque momento. Nel frattempo la CPU ha ripreso l'esecuzione del programma MAIN.

Passo 3

Come già detto più sopra, il software che userete in questo esperimento è costituito da tre routine: INITOC, MAIN e SERVOC. Per una documentazione completa sulla routine MAIN, si veda il capitolo 6. Per quanto, invece, riguarda la routine INITOC e SERVOC, ne vedremo i dettagli qui di seguito.

INITOC: Si tratta di una routine che inizializza come porta d'uscita la Porta Dati A del PIO N. 2 (Nella parola INITOC la lettera O sta per output (uscita), mentre la C sta per Porta C del PIO del Nanocomputer, linee PC0–PC7).

La routine INITOC svolge le seguenti funzioni:

a. Seleziona come modo di interruzione dello Z80 il Modo 2.
b. Inizializza il registro I caricandolo con il byte più significativo dell'indirizzo della Tabella dei Vettori di Interruzione.
c. Carica l'indirizzo della routine di servizio SERVOC nella Tabella dei Vettori di Interruzione.
d. Carica il vettore di interruzione nel PIO (Porta C). Il vettore di interruzione è il codice di identificazione del dispositivo, che il dispositivo PIO porrà sul bus dei dati durante un'operazione di riconoscimento dell'interruzione da parte della CPU. Questo byte viene scritto nella Porta C del PIO (parte di controllo) secondo il formato seguente:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

Il fatto che il bit meno significativo sia 0, ci dice che il chip PIO interpreta tale byte di controllo come vettore dell'interruzione. In questo esperimento, il vettore d'interruzione è 06.
e. Seleziona il Modo 0 come modo di funzionamento del PIO.
f. Per abilitare il funzionamento dell'interruzione scrive nel PIO la Parola di Controllo delle Interruzioni, la quale, per il Modo 0 di funzionamento del PIO, si presenta con il seguente formato:

D7	D6	D5	D4	D3	D2	D1	D0
EI	X	X	X	X	1	1	1

dove EI è il flag di abilitazione delle interruzioni. Se EI viene posto a 1, la Parola di Controllo delle Interruzioni abilita la porta del PIO a generare interruzioni.

Ciò significa che, se il segnale $\overline{CSTB}$ della porta viene attivato da un dispositivo periferico, il PIO attiverà a sua volta il segnale $\overline{INT}$ per richiedere una interruzione della CPU. Notate che l'interruzione viene generata dal fronte positivo del segnale di strobe ($\overline{CSTB}$). La Figura 7-8 illustra la temporizzazione relativa dei diversi segnali per il Modo 0.
Se si azzera il bit EI del PIO, si ha come conseguenza la disabilitazione delle interruzioni per la porta del PIO. In altri termini il PIO non attiverà la linea $\overline{INT}$ quando il segnale $\overline{CSTB}$ va da basso a alto.

g. Pone in uscita un byte iniziale non richiesto dalla periferica, facendo in modo così che il segnale CRDY diventi attivo. Per alcuni dispositivi di uscita, la CPU deve inviare un byte di "attenzione" (wake-up) per (a) attivare la periferica e (b) comunicare alla periferica che può richiedere un byte in uscita abbassando temporaneamente la linea $\overline{CSTB}$. Per periferiche "stupide" che non sono in grado di distinguere tra byte di "attenzione" e byte di dati, la CPU non ha invece bisogno di porre in uscita un byte iniziale. Resta comunque valido il principio che inviare in uscita un byte iniziale non richiesto rappresenta una buona tecnica di programmazione per fissare un insieme coerente di regole operative *per la periferica:*

   (1) LEGGI UN BYTE IN USCITA SOLO SE IL SEGNALE ARDY E' POSTO A UNO.

   (2) RICHIEDI IL BYTE DI DATI SUCCESSIVO ABBASSANDO TEMPORANEAMENTE $\overline{CSTB}$.

   Poichè il byte iniziale non è stato richiesto, la periferica lo ignora. I byte successivi invece verranno richiesti ed "eseguiti" (cioè, stampati, perforati, registrati, ecc.) alla velocità propria della periferica.

SERVOC: Si tratta della routine di servizio dell'interruzione per l'uscita sulla Porta C del PIO N. 2. Nella denominazione SERVOC la lettera O e la C hanno lo stesso significato che hanno nella routine INITOC.

La routine SERVOC ha le seguenti funzioni:

a. Salva lo stato dei registri H, L, A e F della CPU, che sono i soli registri usati dalla routine SERVOC.
b. Pone in uscita il valore corrente del contatore sulla Porta Dati C del PIO. Il valore del contatore, che viene decrementato dalla routine MAIN, viene memorizzato nella locazione ADDL.

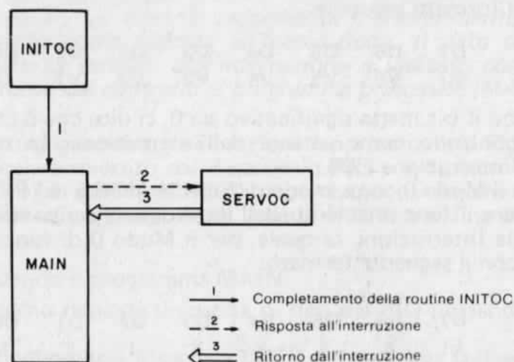


Figura 7-8. Flusso del controllo tra le routine INITOC, MAIN e SERVOC.

c. Ripristina lo stato della CPU.
d. Abilita le interruzioni. Ricordate che le interruzioni mascherabili vengono disabilitate quando la CPU accetta e riconosce una richiesta di interruzione.
e. Cede il controllo alla routine MAIN (istruzione di RETI)
Nella Figura 7-8 è illustrato il flusso del controllo cioè i passaggi, tra le routine INITOC, MAIN e SERVOC.

NOTA: Vorremmo a questo punto fare una breve digressione per illustrare il protocollo di handshake RDY/$\overline{STB}$ del PIO.
Come già detto, quando una periferica abbassa $\overline{STB}$ segnala la richiesta di un nuovo byte di dati. Ne deriva che il primo byte, non richiesto, emesso dalla CPU viene interpretato, in questo contesto, come byte di attenzione (wake up). Oltre alla precedente (interpretazione REQUEST) esiste, però, un altro modo di interpretare il protocollo di handshake del PIO (interpretazione ACKNOWLEDGE), secondo il quale il primo byte non richiesto può essere visto come un dato.
Quando una periferica abbassa $\overline{STB}$, significa che sta confermanto (acknowledge) di aver ricevuto l'ultimo byte.
In questo caso, cioè interpretando l'attivazione di $\overline{STB}$ come una segnalazione di dato ricevuto, la CPU inizia le operazioni fornendo il primo byte di dato e lo scambio procederà poi con una velocità che dipenderà dalla capacità della periferica di ricevere i dati successivamente posti in uscita dalla CPU.
Le due interpretazioni ACKNOWLEDGE (conferma) e REQUEST (richiesta) dei protocolli sono equivalenti con la sola eccezione del significato del primo byte non sollecitato. Nel nostro caso abbiamo preferito l'interpretazione REQUEST semplicemente perchè ci è sembrato più naturale. Certamente altre interpretazioni del protocollo di handshake del PIO sono più adatte alle situazioni specifiche e alle periferiche che avrete occasione di incontrare. In ogni caso siate flessibili e ricordate che, il protocollo rimane costante, anche se esistono più modi per descriverlo.

Passo 4

Iniziate l'esecuzione alla locazione INITOC. Osservate la routine MAIN in esecuzione, cioè la solita visualizzazione di un contatore a decremento. Come noterete, CRDY è attivo (1 logico), come risulta dall'indicatore luminoso acceso, così come sono tutti accesi gli indicatori LM0-LM7, che visualizzano il byte iniziale non richiesto che la CPU ha inviato alla Porta C del PIO.

Passo 5

In questo esperimento, voi stessi rappresentate il dispositivo periferico interfacciato con la Porta C del chip PIO N. 2. Non appena siete pronti per ricevere un byte di dati dovete farne richiesta, attivando la linea $\overline{CSTB}$, cioè premendo il pulsante P0. Cosa notate?

Noterete che la linea CRDY è andata alta e il valore corrente del contatore viene visualizzato sugli indicatori luminosi LM0-LM7.

Passo 6

Tenete ora premuto il pulsante $\overline{P0}$. Verificate che non succede niente. Riportatelo ora nella posizione normale. Che cosa si è verificato?

Rilasciandolo, quasi contemporaneamente gli indicatori si modificano visualizzando il valore corrente del contatore.

Passo 7

Avete notato qualche cambiamento nei LED che visualizzano lo stato delle linee CRDY e $\overline{BINT}$?

Noi non abbiamo osservato nessun cambiamento, nonostante che dalla lettura degli indicatori luminosi si possa dedurre che è stata generata una interruzione. Esaminiamo il diagramma dei tempi del PIO nel Modo 0 e cerchiamo di capire come mai non abbiamo visto nessun cambiamento sulle linee $\overline{BINT}$ e CRDY. La Figura 7-9 presenta il diagramma dei tempi tratto dal *Manuale Tecnico del PIO Z80*. La Figura 7-10 vi mostra la stessa situazione presa da un altro punto di vista.

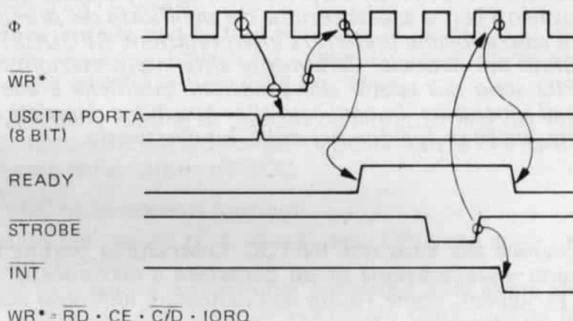


Figura 7-9. Diagramma dei tempi del PIO Z80 nel Modo 0.

A partire dall'attivazione della linea $\overline{STROBE}$ ($\overline{ASTB}$ o $\overline{CSTB}$) si assiste al verificarsi dei seguenti eventi:

1. Con il fronte di salita del segnale di $\overline{STROBE}$, il PIO porta attiva la sua linea $\overline{INT}$, che attraverso un buffer (diventa $\overline{BINT}$) è portata alla CPU, e richiede così una interruzione alla CPU.
2. Con il successivo fronte di discesa del clock, il PIO disattiva la linea READY (ARDY o CRDY), indicando che il dato in uscita non è pronto.
3. La CPU riconosce l'interruzione (segnale $\overline{INTA}$) e corrispondentemente il flip-flop interno al PIO di interruzione pendente viene posto a 0 cosicchè la linea $\overline{INT}$ può ritornare alta (non attiva).
4. Non appena la CPU cede il controllo alla routine di servizio delle interruzioni SERVOC, si ha l'esecuzione di una istruzione di uscita sulla Porta Dati del PIO. I segnali della CPU attivati durante tale ciclo di uscita si combinano fra loro generando il segnale $\overline{WR}$*. (Per maggiori dettagli si veda di seguito).

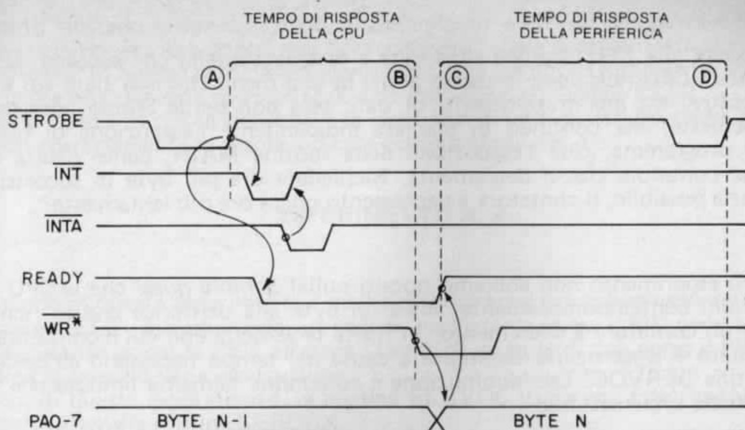


Figura 7-10. Diagramma dei tempi del PIO Z80 nel Modo 1. All'istante A la periferica richiede un byte; nel successivo istante B la CPU inizia il ciclo d'uscita per effettuare l'invio del prossimo byte; all'istante C il PIO fa sapere alla periferica che il byte di uscita è pronto. Infine all'istante D la periferica richiede un altro byte per iniziare il ciclo.

5. In corrispondenza dell'attivazione del segnale $\overline{WR}$*, il PIO acquisisce il dato posto in uscita dalla CPU tramite il bus dei dati D0-D7 e trasferisce tale dato nel registro di uscita del PIO.
6. Con il successivo fronte di discesa del clock, il PIO attiva il segnale READY per indicare che il dato in uscita è finalmente disponibile e pronto per essere letto dal dispositivo esterno.

Se ne deduce, quindi, che la linea $\overline{INT}$ è bassa solo per il tempo necessario alla CPU per sentirla e confermare tale acquisizione inviando al PIO un segnale $\overline{INTA}$ (formato in effetti da $\overline{M1}$ e $\overline{IORQ}$ entrambi attivi). La linea CRDY invece rimane bassa dal momento in cui viene generata l'interruzione fino al momento in cui la routine di servizio della interruzione esegue l'istruzione OUT, in effetti per pochi cicli di istruzione. Il fatto, quindi di non aver osservato nessun cambiamento sui LED $\overline{INT}$ e $\overline{CSTB}$ non dipende dal fatto che tale cambiamento non si sia verificato, ma dal fatto che i nostri occhi non sono in grado di vedere un cambiamento così veloce.

SPIEGAZIONE DEL SEGNALE $\overline{WR}$*: Il segnale WR* è un segnale attivo basso generato all'interno del dispositivo PIO. Va notato che il PIO è dotato di piedini di ingresso solo per $\overline{IORQ}$, $\overline{M1}$ ed $\overline{RD}$, che l'uscita $\overline{WR}$ della CPU Z80 non è collegata al PIO e che il PIO stesso è perciò in grado di dedurre dagli stati logici di $\overline{RD}$, $\overline{IORQ}$, $\overline{CE}$ e C/D se la CPU vi sta effettuando un'operazione di scrittura di un dato.

$\overline{WR}$* è perciò attivo se e solo se:

> ($\overline{RD}$ è NON attivo) E ($\overline{CE}$ è attivo) E (C/D è posto ad 1, in modo da indicare un byte di dati) E ($\overline{IORQ}$ è attivo).

Purtroppo questa relazione viene anche, sovente, scritta così:

$$\overline{WR}^* = \overline{RD} \cdot \overline{CE} \cdot C/D \cdot \overline{IORQ}$$

in una notazione che combina logica positiva e logica negativa. Semplificando, della definizione di $\overline{WR}$* fornita qui sopra in termini di equazione limitatevi a ricordare quali segnali devono essere attivi perchè $\overline{WR}$* sia anch'esso attivo.

Passo 8

Richiedete alla CPU qualche altro byte e osservate quello che succede. Noterete subito che la CPU non pone in uscita nuovi byte a meno che non siate voi a richiederlo. Inoltre, tra due trasferimenti di dati, essa non perde tempo aspettando le vostre richieste, ma continua in maniera indipendente l'esecuzione di una altra parte di programma, cioè l'esecuzione della routine MAIN, come risulta chiaramente dal contatore che si decrementa. Richiedete ora più byte in successione, il più in fretta possibile, Il contatore a decremento opera ora più lentamente?

Nel nostro esperimento non abbiamo notato nulla! *Sembra quasi* che la CPU esegua due funzioni contemporaneamente: invia un byte alla periferica che lo richiede e visualizza un contatore a decremento. In realtà la velocità con cui il contatore viene decrementato è leggermente diminuita a causa del tempo necessario all'esecuzione della routine SERVOC. Tale diminuzione è comunque talmente limitata che risulta a noi del tutto impercettibile.

Passo 9

Resettare il Nanocomputer e apportate quindi le seguenti modifiche alla routine INITOC:

Locazione	Contenuto precedente	Nuovo contenuto
THROW	D3	00
THROW + 1	08	00

Con queste modifiche avete tolto l'istruzione che pone in uscita il byte di wake-up e attiva il segnale CRDY. Iniziate l'esecuzione da INITOC. Cosa notate?

Gli indicatori luminosi LM0-LM7 rimangono spenti, visualizzando così il byte 00. Poichè non è stato ancora posto in uscita nessun byte verso il PIO, il contenuto del registro dei dati in uscita della Porta C del PIO non è definito e potreste quindi leggere un valore diverso. Noterete inoltre che il LED CRDY non è acceso. Se voi, come dispositivo esterno, siete stati progettati per seguire la regola:

LEGGI UN DATO SE, E SOLO, SE CRDY E' ATTIVO

resterete in attesa all'infinito che la linea CRDY diventi alta. C'è una ragione per stabilire sempre una regola di questo tipo? La risposta è affermativa, in quanto tale norma impedisce al dispositivo esterno di leggere lo stesso byte in uscita due volte o di leggere un byte non corretto. Supponiamo, ad esempio, che ci sia stato un lungo ritardo tra la richiesta di un byte da parte della periferica ($\overline{CSTB}$ basso) e il trasferimento del byte dalla CPU al PIO (CRDY attivo). Il segnale CRDY è un importante indicatore per la periferica, poichè, se quest'ultima dovesse leggere il byte dalla porta del PIO *prima* che la linea CRDY *fosse attivata*, essa leggerebbe lo stesso byte in essa contenuto quando $\overline{CSTB}$ era stato portato basso.
Se la periferica stava chiedendo un nuovo byte ed aveva già letto quello precedente, il risultato sarebbe quindi una doppia lettura dello stesso byte.

Abbiamo così due situazioni: da un lato, all'inizio CRDY non è attivo, d'altra parte la periferica deve sempre effettuare una richiesta per far funzionare il meccanismo di trasferimento. E' necessaria perciò, una eccezione alla regola precedente. Una periferica abbastanza "intelligente" può essere predisposta a riconoscere l'eccezione iniziale, continuando poi a seguire rigidamente la regola. Altre periferiche meno

intelligenti richiedono invece l'invio di un byte di wake-up per porre CRDY alto, cosicchè la periferica non debba mai "preoccuparsi" delle eccezioni alla regola.

Poichè voi, in questo caso, venite considerati una periferica intelligente, dovete fare inizialmente una richiesta di un byte di dati abbassando il segnale $\overline{CSTB}$.
Da questo punto in poi la richiesta e la ricezione di byte di dati dovrebbe procedere seguendo correttamente la regola indicata precedentemente.

ESPERIMENTO N. 3

Scopo

Lo scopo di questo esperimento è quello di illustrare il funzionamento del segnale di handshake CRDY, introducendo un ritardo tra l'istante in cui il PIO genera l'interruzione e l'istante in cui la CPU effettua l'uscita di un byte di dati verso il PIO tramite la routine di servizio della interruzione.
Per i fini di questo esperimento, la routine SERVOC usata nell'Esperimento N. 2, è stata sostituita dalla routine SEROCX.

Schema del circuito

In questo esperimento è stato utilizzato lo stesso schema dell'Esperimento N. 2 (Si vedano gli schemi 1A e 1B rispettivamente in Figura 7-5 e 7-6).

Programmi INTOC, MAIN e SEROCX

L'esperimento si basa sulle routine INITOC e MAiN già usate nell'Esperimento N. 2, mentre la routine di servizio della interruzione sarà la routine SEROCX.

Codice oggetto		Codice sorgente	Commenti
C5	SEROCX:	PUSH BC	; salva i registri della CPU
D5		PUSH DE	
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
FD2AE40F		LD IY(ADDL)	; salva il contenuto di ADDL
FDE5		PUSH IY	
DD23	DSX:	INC IX	; aggiorna il puntatore dello stack dei dati
DD23		INC IX	
DD23		INC IX	
00		NOP	; nessuna operazione

Codice oggetto		Codice sorgente	Commenti
DD3600FF		LD (IX+00H),0FFH	; predisponi il tempo di DLOOPX
DD36010A		LD (IX+01H),00AH	; predisponi il tempo di CLOOPX
DD360201	CLOOPX:	LD (IX+02H),01H	; predisponi il tempo di DLOOPX
21E50F		LD HL,ADDH	; punta al buffer del display
ED57		LD A,I	; cerca il valore di IFF2
EA0E06		JP PE,HIGHX	
3600	LOWX:	LD (HL),00H	; valore = 0
1802		JR NEXTX	
3610	HIGHX:	LD (HL),10H	; valore = 1

```
2B         NEXTX:   DEC HL           ; sposta il puntatore del buffer
34                  INC (HL)         ; incrementa ADDL
ED73E20F            LD (DATAL),SP    ; trascrivi SP nel buffer
21B90F              LD HL,LEDL       ; predisponi HL per CONVDI
11E50F              LD DE,ADDH       ; predisponi DE per CONVDI
CD7CFA              CALL CONVDI
CD09F9     DLOOPX:  CALL DISPL
DD3500              DEC (IX+00)      ; temporizzatore per la visualizzazione
20F8                JR NZ,DLOOPX
DD3502              DEC (IX+02)      ; temporizzatore per la visualizzazione
20F3                JR NZ,DLOOPX
DD3501              DEC (IX+01)      ; temporizzazione per la routine di servizio
20CD                JR NZ,CLOOPX
FDE1                POP IY           ; ripristina i registri della CPU
FD22E40F            LD (ADDL),IY     ; ripristina il contenuto di ADDL
3AE40F     OUTX:    LD A,(ADDL)      ; poni in uscita il byte presente
D308                OUT (08H),A      ; in ADDL al momento della interruzione
FDE1                POP IY           ; ripristina i registri della CPU
DDE1                POP IX
F1                  POP AF
E1                  POP HL
D1                  POP DE
C1                  POP BC
FB                  EI               ; abilita le interruzioni
ED4D                RETI             ; ritorna dall'interruzione
```

Passo 1

Questo esperimento può essere considerato la continuazione dell'Esperimento N. 2 Se non avete ancora eseguito l'Esperimento N. 2 vi consigliamo di farlo prima di procedere con l'esecuzione di questo.
La routine SEROCX è molto simile alla routine SERVOC usata nell'Esperimento N. 2 e svolge le seguenti funzioni:

1. Salva lo stato della CPU.
2. Garantisce la "rientrabilità" assegnando un nuovo spazio di dati per i loop del contatore.
3. Genera un ritardo di circa 10 secondi tra due incrementi successivi del contatore.
4. Pone in uscita il valore corrente del contatore verso gli indicatori luminosi.
5. Ripristina lo stato della CPU.
6. Restituisce il controllo alla routine MAIN.

Al fine di sostituire SERVOC con SEROCX come routine di servizio della interruzione, apportate alla Tabella dei Vettori di Interruzione le seguenti modifiche:

Locazione	Contenuto precedente	Nuovo contenuto
INITOC + 10	Byte meno significativo dell'indirizzo di SERVOC	Byte meno significativo dell'indirizzo di SEROCX
INITOC + 11	Byte più significativo dell'indirizzo di SERVOC	Byte più significativo dell'indirizzo di SEROCX

Fate attenzione che nella routine INITOC sia stato ripristinato il byte di wake-up, in modo tale che INITOC si presenti come all'inizio dell'Esperimento N. 2. In particola-

re, controllate che le seguenti locazioni abbiano questi valori:

Locazione	Valore
THROW	D3
THROW + 1	08

Passo 2

Iniziate l'esecuzione a partire da INITOC. Premete il pulsante $\overline{P0}$. Che cosa osservate?

Dovreste notare che:

a. Il contatore sul display della tastiera non viene più decrementato ma incrementato.
b. Il LED CRDY si spegne.
c. Dopo che il contatore ha effettuato 10 incrementi, si verificano simultaneamente i seguenti eventi:
   - CRDY passa a 1 logico e il LED corrispondente si accende.
   - Il valore che il contatore aveva quando è stata generata l'interruzione viene visualizzato sugli indicatori luminosi LM0-LM7.
   - Il controllo della CPU viene ceduto alla routine MAIN, come risulta evidente dal contentuto del puntatore dello stack che ritorna a 0F00 e dal contatore che riprende a decrementare.

E' importante osservare che c'è stato un ritardo significativo tra la richiesta di un nuovo byte (quando avete premuto il pulsante $\overline{P0}$) e l'arrivo di quest'ultimo. Durante questo intervallo, il segnale CRDY è stato disattivato, indicando così che la periferica non può ancora leggere la porta del PIO, ma deve attendere. Per interagire correttamente con il dispositivo PIO, la periferica dovrà quindi essere in grado di effettuare questa operazione di attesa.

ESPERIMENTO N. 4

Scopo

Lo scopo di questo esperimento è quello di illustrare il funzionamento del PIO nel Modo 1.
Una porta del PIO che operi nel Modo 1 è analoga ad una porta di ingresso realizzata con buffer e latch convenzionali. Esiste però un'altra caratteristica propria del PIO, cioè la possibilità di acquisire dati in interruzione utilizzando i due segnali di handshake $\overline{STB}$ e RDY.

Schema del circuito (Figura 7-11)

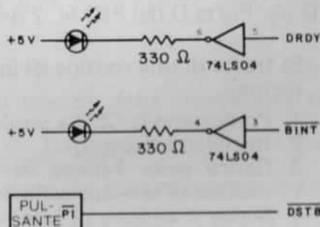


Figura 7-11. Schema N. 2. Circuito per l'Esperimento PIO nel Modo 1.

Programmi MAIN, INITID, SERVID e SERVI

Oltre alla routine MAIN, in questo esperimento vengono usati i seguenti programmi:

Codice oggetto		Codice sorgente	Commenti
ED5E	INITID:	IM2	; modo di interruzione 2
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; predisponi il registro di interruzione
FD211F04		LD IY,SERVID	; indirizzo routine di servizio dell'interru- ; zione
FD22080F		LD (TABLE+08H),IY	; trasferiscilo nella tabella dei vettori
3E08		LD A,08H	; carica il vettore di interruzione
D30B		OUT (0BH),A	
08		EX AF,AF'	; predisponi il formato per CONVDI
3E40		LD A,40H	
08		EX AF,AF'	
3E4F		LD A,4FH	; predisponi il modo del PIO
D30B		OUT (0BH),A	
3E87		LD A,87H	; abilita le interruzioni del PIO
D30B		OUT (0BH),A	
DB09		IN A,(09H)	; inizializza DRDY
C3C302		JP MAIN	

Codice oggetto		Codice sorgente	Commenti
C5	SERVID:	PUSH BC	
0E09		LD C,09H	; interruzione della PORTA D
C33104		JP SERVI	

Nota: Per la routine SERVI si veda l'Appendice B pag. 427 e 428

Passo 1

Collegate il circuito come indicato nello schema. Notate che i due LED sono collegati in modo tale da visualizzare lo stato di $\overline{DSTB}$ e DRDY. Ricordate inoltre che in questo esperimento userete le linee PD0-PD7 della Porta Dati B del PIO N. 2.

Passo 2

INITID e SERVID sono le nuove routine di questo esperimento. (La I sta per Input, ingresso, e la D per Porta D del PIO N. 2 del Nanocomputer).

INITID: Si tratta di una routine di inizializzazione che effettua le seguenti operazioni:

1. Predispone la CPU a servire le interruzioni in Modo 2.
2. Inizializza il registro I.
3. Carica nella Tabella dei Vettori di Interruzione l'indirizzo della routine di servizio della interruzione SERVID.
4. Scrive il vettore di interruzione nella Porta di controllo B del PIO N. 2 (Porta D per il Nanocomputer).

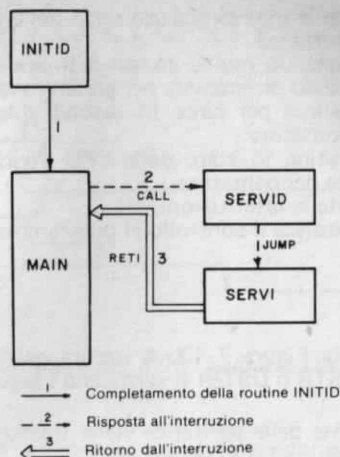


Figura 7-12. Flusso del controllo per l'Esperimento con il PIO in modo di ingresso.

5. Predispone la Porta D del PIO N. 2 perchè operi nel Modo 1 (modo di ingresso). A tal fine scrive il seguente byte nell'indirizzo 0B, porta di Controllo D:

0 1 0 0	1 1 1 1
4	F

Notate che il gruppo di 4 bit meno significativi indica che si tratta di un byte di selezione del modo, mentre i due bit più significativi specificano il modo selezionato.

6. Abilita l'interruzione per la Porta D: ciò viene effettuato scrivendo in uscita il byte seguente all'indirizzo 0B:

1 0 0 0	0 1 1 1
8	7

Notate che il gruppo di 4 bit meno significativi sta ad indicare che si tratta di un byte di controllo dell'interruzione. Nel Modo 1, l'unico bit che viene usato è il bit D7, bit di abilitazione/disabilitazione delle interruzioni. Quando tale bit è posto a uno le interruzioni sono abilitate. Ricordate che nel PIO il flip-flop di abilitazione interruzioni è completamente indipendente dai flip-flop IFF1 e IFF2 interni alla CPU Z80.

7. Inizializza DRDY a livello logico 1 (attivo) inviando un byte di wake-up al PIO N. 2-B sulle linee PD0-7, indirizzate come porta 09.

SERVID: Si tratta di una routine di servizio della interruzione che effettua le seguenti operazioni:

1. Salva lo stato dei registri della CPU.
2. Memorizza il valore corrente del contatore memorizzato nella locazione ADDL.
3. Legge il byte di dato dalla Porta Dati D del PIO N. 2.

4. Carica il byte così acquistato nella locazione ADDL perchè sia poi visualizzato.
5. Assegna un nuovo spazio dati per i byte di temporizzazione usati in questo programma per garantire la rientrabilità.
6. Visualizza per circa 10 secondi il byte acquisito invece del valore del contatore.
7. Ripristina lo stato della CPU e ricarica nella locazione ADDL il valore del contatore.
8. Abilita le interruzioni.
9. Restituisce il controllo al programma MAIN.

Passo 3

Fate riferimento alla Figura 7-13. A partire dall'istante in cui diventa attivo il segnale di $\overline{STROBE}$ ($\overline{BSTB}$ o $\overline{DSTB}$) si verificano i seguenti eventi:

1. Il dato proveniente dalla periferica viene trasferito dal PIO in un registro interno di ingresso dati.
2. Il fronte di salita del segnale $\overline{STROBE}$ fa sì che il PIO attivi la linea INT, richiedendo una interruzione alla CPU Z80.
3. Con il successivo fronte di discesa del clock, il segnale READY viene portato basso (non attivo), per indicare che la CPU non ha ancora letto il byte inviato dalla periferica. Quindi, anche per le operazioni di ingresso, il segnale READY è essenziale, in quanto indica alla periferica che il registro di ingresso del PIO è pieno e non deve essere ulteriormente caricato finchè la CPU non lo ha letto.
4. La CPU cede il controllo alla routine di servizio della interruzione.
5. All'interno della routine di servizio della interruzione viene eseguita una istruzione IN, lettura (ingresso) di un dato dalla Porta Dati del PIO, che attiva i segnali:

   $$\overline{RD}, \overline{IORQ} \text{ e } \overline{CE}$$

   e porta alta la linea C/D. Di conseguenza anche $\overline{RD}^*$ diventa attivo. Alla fine dell'operazione di lettura viene riattivato il segnale READY.

NOTA: La routine SERVID non viene memorizzata in locazioni di memoria contigue. Essa è costituita da due sezioni (segmenti) di codice: il primo segmento è formato da 6 byte il cui inizio è etichettato con SERVID. Questa parte serve a memorizzare il contenuto della coppia di registri BC nello STACK, caricando quindi il registro C con l'indirizzo 0CH della Porta Dati D del PIO N. 2

Il controllo viene quindi ceduto alla locazione di memoria SERVI, punto d'inizio del secondo segmento di codice. E' interessante osservare qui due fatti interessanti: 1) il segmento di codice che inizia a SERVI è RIENTRANTE; 2) l'indirizzo della porta viene trasferito a questo segmento di codice tramite il registro C.

Saremo quindi in grado di usare il segmento di codice che inizia a SERVI per qualsiasi porta che desideriamo leggere. Sostituendo poi l'istruzione NOP alla locazione ENABG con l'istruzione che abilita il flip-flop dell'interruzione dello Z80 (istruzione EI — codice esadecimale FB) lo stesso segmento di codice SERVI, in occasione, potrà essere ancora interrotto e utilizzato, per servire una nuova richiesta di una qualsiasi porta.

In particolare, per ogni porta di ingresso disponibile avremo una routine di servizio dell'interruzione che leggerà dalla Porta. A questo punto possediamo un meccanismo che gestisce in modo assai efficiente le routine di servizio interruzione di questo tipo. Ogni routine di servizio inizierà con una sezione di "preparazione" formata da 6 byte simili ai primi 6 byte della routine SERVID. Il controllo verrà poi ceduto al segmento di codice SERVI, che è una routine rientrante (Re-entrant), e può perciò essere utilizzata contemporaneamente da più dispositivi esterni. A questo punto dovreste esaminare e cercare di capire bene le routine SERVID, SERVIC, SERVIE e SERVIF.

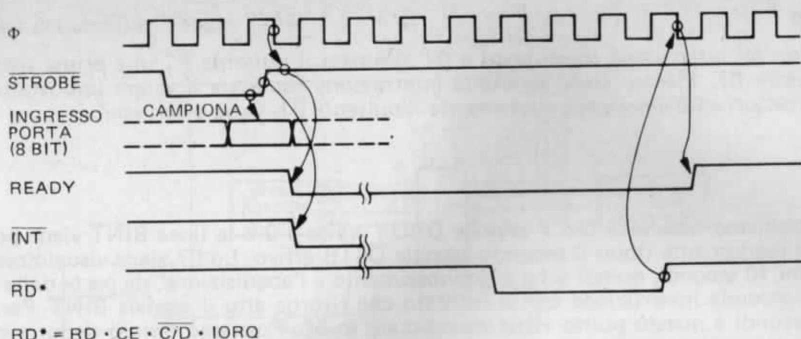


Figura 7-13. Diagramma dei tempi per il PIO nel modo di ingresso (Modo 1).

6. Quando la linea READY è ritornata attiva, la periferica può chiedere alla CPU di acquisire un nuovo byte.

Passo 4

Ponete 07 sugli interruttori SW0-SW7. Iniziate l'esecuzione da INITID. A questo punto dovreste vedere la solita visualizzazione della routine MAIN e il LED BINT acceso. Il LED DRDY è acceso perchè la routine INITID ha letto un byte di wake-up. Tenete premuto il pulsante P1. Non dovreste osservare nessun cambiamento né sul display né sui LED. Lasciate andare il pulsante. Cosa osservate?

Noterete che sul display il conteggio si arresta (nel nostro caso si è fermato su 03) e che il valore impostato sugli interruttori, 07, viene visualizzato per circa 10 secondi al posto del contatore. Durante questo intervallo il display indica che IFF2 (e quindi IFF1) è stato posto a zero, mostrando così che le interruzioni mascherabili sono state disabilitate. Il puntatore dello stack passa da 0F00 a 0EEC e, come potete dedurre, sono stati perciò posti nello stack 12 byte.

NOTA: Può succedere (ma è piuttosto improbabile) che notiate lo stack passare a 0EF0 invece che a 0EEC? Perchè?

Passo 5

Modificate il byte di ingresso sugli interruttori e generate un altro impulso DSTB. Dovreste osservare che la nuova configurazione degli interruttori compare sul display e vi rimane per circa 10 secondi, esattamente come prima. Ripetete la procedura per diversi valori di ingresso finchè non siete in grado di comprendere il funzionamento dei segnali di handshake DSTB e DRDY.

Passo 6

Con gli interruttori di ingresso impostati su 07, generate un impulso DSTB. Mentre l'interruzione viene servita, premete il pulsante DSTB di nuovo. Cosa notate?

Noterete che il segnale DRDY è stato disattivato e che è stato visualizzato 07 per il doppio della durata di una singola interruzione.

Passo 7

Con gli interruttori logici posti a 07, premete il pulsante $\overline{\text{P1}}$ una prima volta per acquisire 07. Mentre viene servita la interruzione cambiate il valore impostato sugli interruttori a 00 e premete nuovamente il pulsante $\overline{\text{P1}}$. Cosa osservate?

Noi abbiamo osservato che il segnale DRDY passa a 0 e la linea $\overline{\text{BINT}}$ viene portata bassa esattamente dopo il secondo segnale $\overline{\text{DSTB}}$ attivo. Lo 07 viene visualizzato per i primi 10 secondi, quindi si ha il riconoscimento e l'acquisizione, da parte della CPU, della seconda interruzione con il risultato che ritorna alto il segnale $\overline{\text{BINT}}$. Per circa 10 secondi a questo punto viene visualizzato lo 00. Possiamo quindi dedurre che le due interruzioni sono state servite in modo SERIALE, cioè una dopo l'altra. Perchè la seconda interruzione non è stata "innestata" nella prima? Perchè cioè la seconda interruzione non ha interrotto il servizio della prima?

La risposta è che le interruzioni della CPU vengono disabilitate durante l'esecuzione della routine SERVID. Solo con la penultima istruzione le interruzioni vengono riabilitate. Il PIO ricorda la seconda interruzione (che diventa così "pendente") grazie ad un flip-flop interno: quando infine la CPU riabilita le interruzioni mascherabili, la seconda interruzione viene individuata, accettata e servita.

Passo 8

Quante interruzioni può mantenere pendenti il PIO in attesa che la CPU sia in grado di accettarne di nuove?
Verificatelo generando più segnali $\overline{\text{DSTB}}$ attivi in successione veloce. Quante di queste interruzioni vengono servite?

Noi abbiamo osservato che sono riconosciute e servite solo due richieste che portano perciò alla visualizzazione del byte 00 per 20 secondi. Quindi, ricordando il flip-flop di interruzione pendente che avete montato nell'Esperimento N. 1 del Capitolo 6, possiamo dire che il PIO è dotato di un circuito interno simile, in grado di ricordare una sola interruzione pendente.

ESPERIMENTO N. 5

Scopo

Lo scopo di questo esperimento è di illustrare il funzionamento del PIO nel Modo 2, cioè nel Modo Bidirezionale. Tale modo è disponibile solo per la Porta A: i segnali di handshake della Porta A vengono usati per controllare l'uscita dei dati, mentre i segnali di handshake della Porta B vengono usati per l'ingresso. Quando la Porta A funziona nel Modo 2, la Porta B *deve* essere programmata per il Modo 3. Ricordate che la Porta A del PIO N. 2 è detta Porta C, mentre la Porta B è indicata con il nome di Porta D.

Schema del circuito (Figura 7-14)

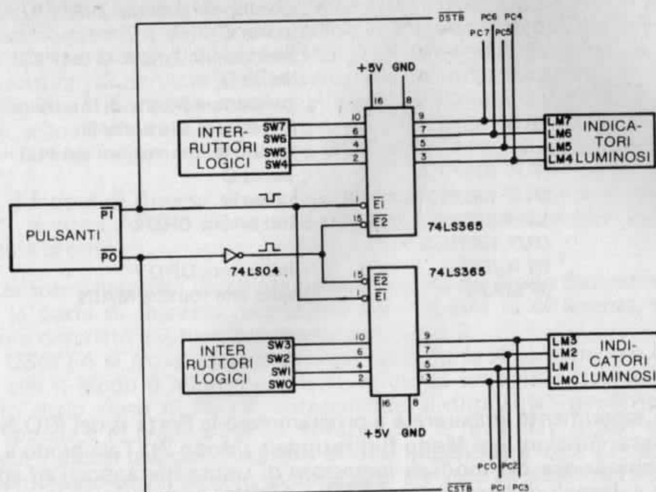


Figura 7-14. Schema N. 3. Circuito per l'esperimento con il PIO nel modo 2.

Programmi: MAIN, SERVOC, SERVIC e INITPB

Oltre alle routine MAIN e SERVOC, già viste nei precedenti esperimenti, in questo esperimento verranno usati i seguenti programmi:

Codice oggetto		Codice sorgente	Commenti
C5	SERVIC:	PUSH BC	; salva BC
0E08		LD C,08H	; interruzione della PORTA C
C33104		JP SERVI	

Codice oggetto		Codice sorgente	Commenti
ED5E	INITPB:	IM2	; modo 2 di interruzione dello Z80
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; predisponi il registro di interruzione
FD21E803		LD IY,SERVOC	; indirizzo routine di servizio per l'uscita dati
FD22060F		LD (TABLE+06H),IY	; metti l'indirizzo nella tabella dei vettori
FD211904		LD IY,SERVIC	; indirizzo routine di servizio per l'ingresso ; dati
FD220A0F		LD (TABLE+0AH),Y	; metti l'indirizzo nella tabella dei vettori
3E06		LD A,06H	; carica il vettore di interruzione
D30A		OUT (0AH),A	; relativo alla porta C
3E0A		LD A,0AH	; carica il vettore di interruzione
D30B		OUT (0BH),A	; relativo alla porta D
08		EX AF,AF'	; predisponi il formato per CONVDI

```
3E40      LD A,40H        ;
08        EX AF,AF'       ;
3E8F      LD A,8FH        ; predisponi il modo 2 del PIO
D30A      OUT (0AH),A     ; porta C
3ECF      LD A,0CFH       ; predisponi il modo 3 del PIO
D30B      OUT (0BH),A     ; porta D
3EFF      LD A,0FFH       ; predisponi il byte di maschera
D30B      OUT (0BH),A     ; necessario alla porta D
3E87      LD A,87H        ; abilita le interruzioni del PIO
D30A      OUT (0AH),A     ; porta C
D30B      OUT (0BH),A     ; porta D
3EFF      LD A,0FFH       ; inizializza CRDY
D308      OUT (08H),A
DB08      IN A,(08H)      ; inizializza DRDY
C3C302    JP MAIN         ; salta alla routine MAIN
```

Passo 1

Con questo esperimento imparerete a programmare la Porta A del PIO N. 2 (Porta C del PIO) perchè funzioni nel Modo Bidirezionale (Modo 2). Tale modo è essenzialmente una combinazione dei modi di ingresso e di uscita sovrapposti ed applicati ad una sola porta. I segnali di handshake $\overline{ASTB}$ e ARDY sincronizzano le operazioni di uscita dei byte, mentre i segnali $\overline{BSTB}$ e BRDY coordinano le operazioni di acquisizione. Le regole fondamentali per il funzionamento dei segnali di handshake sono molto simili a quelle già presentate per il Modo 0 e per il Modo 1, con una sola eccezione molto importante.
La Figura 7-15 illustra il diagramma dei tempi relativo al funzionamento nel Modo 2. A partire dall'istante in cui diventa attivo il segnale $\overline{ASTB}$, si verificano i seguenti eventi:

1. Mentre $\overline{ASTB}$ (in questo esperimento $\overline{CSTB}$) è attivo, la Porta dei dati A (in questo esperimento, le linee PC0-PC7) conserva il valore corrente del registro di uscita della Porta A. La periferica deve leggere questo byte mentre $\overline{ASTB}$ è attivo perchè il byte viene tolto dalle linee dati della Porta A non appena $\overline{ASTB}$ diventa non attivo.
2. Con il fronte di salita $\overline{ASTB}$, il PIO genera una richiesta di interruzione alla CPU Z80.
3. Il segnale ARDY (in questo esperimento, CRDY) viene disattivato, così indi-

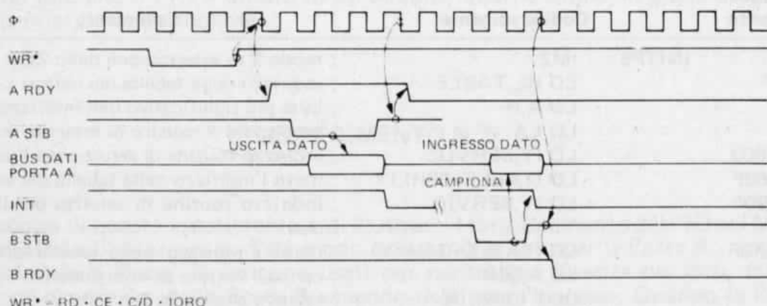


Figura 7-15. Diagramma dei tempi del PIO nel modo bidirezionale (Modo 2).

cando che il prossimo byte di uscita non può ancora essere letto dalla periferica.

4. La CPU sente la linea di interruzione ($\overline{INT}$) bassa, e durante il ciclo di richiesta/riconoscimento della interruzione, il PIO pone sul bus dei dati della CPU gli otto bit meno significativi dell'indirizzo, della tabella dei vettori, che punta alla routine di servizio dell'interruzione relativa all'*uscita*. Il PIO utilizza il vettore di interruzione relativo all'uscita poichè la linea $\overline{ASTB}$ è stata posta bassa, segnalando così la richiesta di un altro byte in uscita.
5. Tale routine esegue un'istruzione OUT sulla Porta A, attivando così il segnale $\overline{WR}$*.
6. Con il fronte di discesa successivo del clock dopo il fronte di salita del segnale $\overline{WR}$*, la linea ARDY viene attivata, indicando che il nuovo byte da porre in uscita è pronto.

La sequenza sopra descritta illustra la parte di uscita del Modo Bidirezionale. Prima di esaminare la parte di ingresso, vorremmo farvi notare la differenza sostanziale tra quanto sopra descritto e il funzionamento nel Modo 0.
Il dato in USCITA si trova sul bus dei dati della Porta A solo mentre $\overline{ASTB}$ è attivo.
Ricordate che il Modo 0 mantiene il byte in uscita sulle linee dati della porta senza tener conto dello stato di $\overline{ASTB}$, consentendo al dispositivo periferico di leggere il byte non appena è in grado di farlo. Il fatto che le linee dati della Porta A vengano usate sia per l'uscita che per l'ingresso di dati impedisce invece di occupare le linee in modo continuato con il dato in uscita, come è invece possibile nel Modo 0.
Quindi i dati vengono abilitati su tali linee solo quando la periferica richiede esplicitamente un altro byte, cioè quando $\overline{ASTB}$ è attivo. Questo fatto ha delle conseguenza particolarmente importanti rispetto alla logica di richiesta di un byte in uscita:

a. Per acquisire un nuovo byte la periferica deve richiederlo esplicitamente.
b. Il byte letto dalla periferica è il byte che la CPU ha trasferito al PIO durante la routine di servizio interruzione richiesta quando il precedente byte è stato letto.
c. La periferica sta sempre un byte "indietro" rispetto alla CPU.

Esaminiamo ora la sequenza di eventi legati alle operazioni di ingresso (lettura) di un byte dalla periferica.
A partire dall'istante in cui diventa attivo il segnale $\overline{BSTB}$, si verificano i seguenti eventi:

1. Il dato da acquisire viene presentato sulle linee dati della Porta A con il fronte di discesa del segnale $\overline{BSTB}$ (in questo esperimento, $\overline{DSTB}$).
2. Mentre $\overline{BSTB}$ è attivo (basso), il PIO legge il byte in ingresso e genera una richiesta di interruzione alla CPU.
3. Con il fronte di discesa successivo del clock, il PIO disattiva il segnale BRDY (in questo esperimento DRDY), per indicare alla periferica che la porta di ingresso del PIO è piena e quindi che non è possibile ricevere un altro byte.
4. La CPU riconosce la linea $\overline{INT}$ attiva ed esegue un ciclo di richiesta/riconoscimento dell'interruzione.
5. Quando riceve la conferma del riconoscimento della interruzione, ($\overline{INTA}$ attivo cioè quando sia $\overline{M1}$ che $\overline{IORQ}$ sono attivi) il PIO pone sul bus dei dati della CPU il byte meno significativo dell'indirizzo, della tabella dei vettori di interruzione, che punta alla routine di servizio dell'interruzione relativa all'*ingresso*.
6. La CPU cede il controllo a tale routine che esegue un'istruzione di ingresso IN per leggere il dato contenuto nel registro (dati) di ingresso della Porta A del PIO.
7. Durante il ciclo di lettura del PIO, il segnale $\overline{RD}$* è posto basso (attivo). Con il fronte di salita del segnale $\overline{RD}$*, la linea BRDY viene attivata, indicando che il byte inviato è stato letto dalla CPU e la periferica può quindi passare all'invio del byte successivo.

Passo 2

In questo esperimento verranno usati i programmi INITPB, MAIN, SERVOC e SERVIC. Dei programmi MAIN e SERVOC abbiamo già parlato nel corso del Capitolo 6 e nell'Esperimento 1 di questo Capitolo rispettivamente.

INITPB: E' questa una routine di inizializzazione che svolge le seguenti funzioni:

a. Definisce per la CPU il Modo 2 di Interruzione.
b. Inizializza il registro I.
c. Carica l'indirizzo di SERVOC nella Tabella dei Vettori di Interruzione.
d. Carica l'indirizzo di SERVIC nella Tabella dei Vettori di Interruzione.
e. Carica il Vettore di Interruzione della Porta C del PIO. Poichè i segnali di handshake della Porta C controllano l'uscita dei byte, tale vettore di interruzione punta alla routine SERVOC.
f. Carica il Vettore di Interruzione della Porta D del PIO. Poichè i segnali di handshake della Porta D controllano l'ingresso dei byte, tale vettore di interruzione punterà alla routine SERVIC.
g. Programma la Porta C del PIO per il Modo 2 di funzionamento.
h. Programma la Porta D del PIO per il Modo 3 di funzionamento. Poichè le linee di handshake della Porta D vengono usate dalla Porta C, la Porta D può essere programmata solo in tale modo.
i. Specifica il byte di maschera della Porta D. Si tratta di un'operazione necessaria quando si opera in Modo 3, anche se non verrà usato.
j. Abilita le interruzioni per le Porte C e D.
k. Inizializza CRDY e DRDY.

SERVIC: Si tratta di una routine di servizio delle richieste di interruzione generate dall'attivazione del segnale $\overline{DSTB}$ del PIO. La routine SERVIC è identica alla routine SERVID con la sola eccezione che l'ingresso avviene tramite la Porta C, anzichè tramite la Porta D. Per maggiore comodità, vi diamo qui di seguito le funzioni principali svolte dalla routine SERVIC:

a. Acquisizione del dato posto sugli interruttori collegati alla Porta C del PIO.
b. Visualizzazione sull'unità tastiera/display del Nanocomputer del byte acquisito, per 10 secondi, al posto del contatore.
c. Restituzione del controllo alla routine MAIN, che continua a decrementare il contatore.

Vediamo ora come le routine SERVIC e SERVOC interagiscono nell'esecuzione delle loro funzioni di ingresso e uscita.
La Figura 7-16 illustra proprio questa interazione.
L'attivazione del segnale $\overline{DSTB}$ fa sì che avvengano i trasferimenti di dati I e II.

TRASFERIMENTO I : Il dato impostato sugli interruttori viene acquisito dal registro di ingresso della Porta C del PIO e viene generata una interruzione.



Figura 7-16. Trasferimenti di dati relativi alle routine SERVIC e SERVOC.

TRASFERIMENTO II : La routine SERVIC acquisisce il dato d'ingresso contenuto nel registro d'ingresso della Porta C del PIO e lo memorizza nella locazione di memoria ADDL per la sua successiva visualizzazione.

L'attivazione del segnale $\overline{CSTB}$ fa sì che avvengano i trasferimenti di dati III e IV:

TRASFERIMENTO III : Il contenuto del registro di uscita del PIO viene posto sulle linee dati della Porta C per essere letto dal dispositivo periferico.

TRASFERIMENTO IV : Sul fronte di salita del segnale $\overline{CSTB}$, il PIO genera una richiesta di interruzione verso la CPU, la quale riconosce l'interruzione e cede il controllo alla routine SERVOC. Quest'ultima pone in uscita il contenuto della locazione di memoria ADDL sul registro di uscita della Porta C del PIO. Questo byte non verrà letto dalla periferica finchè essa non avrà fatto esplicita richiesta di un nuovo byte attivando la linea $\overline{CSTB}$.

Il flusso del controllo tra le routine INITPB, MAIN, SERVOC e SERVIC è stato schematizzato nella Figura 7-17.

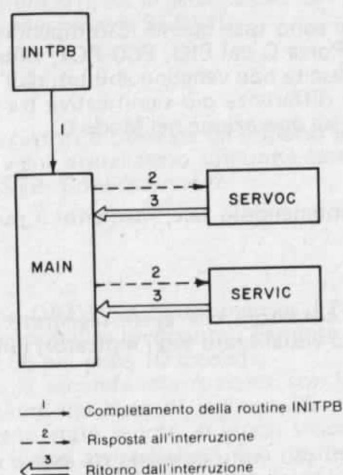


Figura 7-17. Flusso del controllo tra le routine INITPB, MAIN, SERVOC e SERVIC.

Passo 3

Collegate il circuito, necessario per questo esperimento, come indicato nello Schema N. 3 (Figura 7-14). Notate che il pulsante $\overline{P0}$ richiede al PIO l'uscita di un byte, mentre il pulsante $\overline{P1}$ richiede l'ingresso di un byte. I pulsanti vengono collegati ai buffer 74LS365 in modo che il dato presente sugli interruttori sia presentato sulle linee dati della Porta A solo se $\overline{P1}$ è attivo (0 logico) e $\overline{P0}$ non è attivo (1 logico). L'attivazione del solo $\overline{P1}$ produce un impulso di $\overline{DSTB}$ attivo, ma non abilita i buffer. Inoltre, l'attivazione simultanea di entrambi i pulsanti, per altro non consigliabile, non abilita i buffer.

Passo 4

Iniziate l'esecuzione da INITPB. Noterete che:

a. Sia CRDY che DRDY sono attivi (alti)
b. Gli indicatori luminosi (di uscita) sono tutti spenti.
c. La routine MAIN è in fase di esecuzione con il contatore che si decrementa ad una velocità di due decrementi al secondo.

Portate ora la linea $\overline{CSTB}$ a 0 premendo il pulsante $\overline{P0}$ e tenendolo premuto. Che cosa osservate?

Noi abbiamo osservato che gli indicatori luminosi LM0-LM7 visualizzano il byte di wake-up (o di inizializzazione) FF. Lasciate ora ritornare la linea $\overline{CSTB}$ a 1 rilasciando il pulsante. Eseguendo questa operazione prendete nota del valore del contatore presente sul display del Nanocomputer.
Con il fronte di salita del segnale $\overline{CSTB}$ viene generata una interruzione, mentre il controllo della CPU viene trasferito alla routine SERVOC che pone in uscita il valore corrente del contatore sulla Porta C (uscita) del PIO. Cosa vedete ora sugli indicatori luminosi?

Noi abbiamo osservato che sono tutti spenti. Ciò dipende dal fatto che $\overline{CSTB}$ non è attivo. Le linee dati della Porta C del PIO, PC0-PC7, rimangono in uno stato di alta impedenza finchè i dati in uscita non vengono abilitati dall'attivazione di $\overline{CSTB}$. Ricordate che questa è la differenza più significativa tra l'emissione di un byte nel Modo Bidirezionale e la stessa operazione nel Modo 0.

Passo 5

Ponete $\overline{CSTB}$ basso, mantenendolo tale, mediante il pulsante $\overline{P0}$. Che cosa osservate?

Noterete che il valore del contatore che avete registrato quando avete generato il primo impulso $\overline{CSTB}$, risulta visualizzato sugli indicatori luminosi LM0-LM7.

Passo 6

Ripetete i passi precedenti più volte in modo da essere sicuri di aver capito il rapporto tra il byte in uscita *corrente* e il valore del contatore nel momento in cui si verifica l'interruzione *precedente*.

Passo 7

Impostate 03 sugli interruttori, quindi portate $\overline{DSTB}$ basso, e mantenetelo tale, mediante il pulsante $\overline{P1}$. Che cosa osservate?

Noi abbiamo osservato che sugli indicatori luminosi compare immediatamente 03 mentre il display del Nanocomputer rimane immutato, cioè il contatore continua a decrementare, indicando che la routine MAIN controlla ancora la CPU.

Passo 8

Lasciate che il pulsante $\overline{P1}$ ritorni nella sua posizione normale, provocando un fronte di salita sul segnale $\overline{CSTB}$.
Viene così generata una interruzione che trasferisce il controllo della CPU alla routine SERVIC. A questo punto dovreste osservare che, sul display, al posto del contatore, viene visualizzato il byte 03, che vi rimane per circa 10 secondi.

Passo 9

Premete una volta il pulsante $\overline{P0}$ e osservate attentamente il LED che controlla il segnale CRDY. Notate un tremolio?

Noi non lo abbiamo notato anche se sappiamo che CRDY va basso per tutto l'intervallo di tempo che intercorre tra la generazione dell'interruzione e l'esecuzione dell'istruzione OUT da parte della routine SERVOC.
Premete ora $\overline{P1}$ e osservate attentamente il LED che controlla DRDY. Non si dovrebbe notare nessun cambiamento visibile, anche se sappiamo che DRDY va basso per l'intervallo di tempo che intercorre tra la generazione dell'interruzione e l'esecuzione dell'istruzione IN da parte della routine SERVIC.

Passo 10

Impostate 05 sugli interruttori e generate un impulso sulla !inea $\overline{DSTB}$, premendo il pulsante $\overline{P1}$; mentre 05 viene visualizzato sull'unità tastiera/display del Nanocomputer, premete di nuovo $\overline{DSTB}$. Cosa osservate?

Noi abbiamo osservato che DRDY va basso, mentre il PIO memorizza una interruzione in attesa di essere riconosciuta (pendente). Durante il servizio della prima interruzione, viene visualizzato 05 per circa 10 secondi.
Viene quindi riconosciuta la seconda interruzione, con la conseguente riattivazione di DRDY e la visualizzazione del byte di ingresso 05 per altri 10 secondi. Notate che queste interruzioni sono state servite in modo sequenziale (una dopo l'altra) e non in modo "innestato" (l'una dietro l'altra). Ciò risulta in modo evidente non solo dal comportamento del segnale DRDY, ma anche dal valore visualizzato del puntatore di stack.

Passo 11

Premete più volte il pulsante $\overline{P0}$, cambiando così più volte lo stato di $\overline{CSTB}$. Notate qualche cambiamento sulla linea CRDY?

Noi non abbiamo notato nessun cambiamento nel LED che controlla il segnale CRDY, che sembrava rimanere fermo a 1. Ciò dipende dal fatto che la routine di

servizio dell'interruzione viene eseguita più velocemente di quanto voi possiate premere il pulsante.
Ricordate inoltre che la routine SERVOC non ha il loop da 10 secondi come la routine SERVIC.

Passo 12

Impostate 06 sugli interruttori, quindi premete il pulsante $\overline{P1}$ una prima volta per porre in ingresso il byte. Come rilasciate il pulsante $\overline{P1}$, annotate il valore del contatore. Ora, mentre 06 è visualizzato sul display, premete P0 per richiedere un byte in uscita. Che cosa osservate?

Noterete che stavolta CRDY passa a 0, indicando un ritardo tra la richiesta di un byte in uscita e il suo servizio.
Questo ritardo è dovuto alla eccessiva durata dell'esecuzione della routine SERVIC e al fatto che, durante l'esecuzione di quest'ultima, le interruzioni vengono disabilitate.
Una volta che l'esecuzione della routine SERVIC è stata completata, viene servita l'interruzione pendente e riattivata quindi la linea CRDY. Notate che per osservare il byte in uscita che avete appena richiesto premendo il pulsante P0, dovete tenere $\overline{CSTB}$ bassa. Tale byte dovrebbe essere uguale al valore del contatore che vi eravate annotati prima.

ESPERIMENTO N. 6

Scopo

Lo scopo di questo esperimento è quello di illustrare il funzionamento del chip PIO nel Modo 3, detto anche Modo di Controllo. Tale modo è alquanto diverso da quelli presentati finora, in quanto non utilizza segnali di handshake, ed è stato studiato per l'ingresso/uscita di un bit piuttosto che di byte.

Schema del circuito (Figura 7-18)

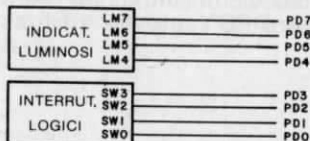


Figura 7-18. Schema N. 4. Circuito per l'esperimento PIO in modo 3.

Programmi: MAIN, INITPM e SERVM

Oltre alla routine MAIN, in questo esperimento verranno usate le seguenti routine:

Codice oggetto	Codice sorgente		Commenti
ED5E	INITPM:	IM2	; modo 2 dell'interruzione dello Z80
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; predisponi il registro di interruzione
FD210505		LD IY,SERVM	; indirizzo della routine di servizio
FD220C0F		LD (TABLE+0CH),IY	; inserisci l'indirizzo nella tabella dei vettori
3E0C		LD A,0CH	; predisponi il vettore di interruzione
D30B		OUT (0BH),A	; relativo alla porta D
08		EX AF,AF'	; predisponi il formato per CONVDI
3E40		LD A,40H	
08		EX AF,AF'	
3ECF		LD A,0CFH	; predisponi il modo 3 per la porta D
D30B		OUT (0BH),A	
3E0F		LD A,0FH	; definisci le linee di ingresso per la
D30B		OUT (0BH),A	; porta D
3E97	CWORD:	LD A,97H	; predisponi la parola di controllo delle in- ; terruzioni
D30B		OUT (0BH),A	
3EFC		LD A,0FCH	; controlla PB0, PB1
D30B		OUT (0BH),A	
0E09		LD C,(09H)	; spegni gli indicatori luminosi
3E00		LD A,00H	
ED79		OUT (C),A	
C3C302		JP MAIN	

Codice oggetto	Codice sorgente		Commenti
C5	SERVM:	PUSH BC	; salva il registri della CPU
D5		PUSH DE	
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
FD2AE40F		LD IY,(ADDL)	; salva il contenuto di (ADDL)
FDE5		PUSH IY	
0E09		LD C,09H	; leggi della porta C del PIO
ED78		INC A,(C)	
E60F		AND 0FH	; azzera la parte più significativa del byte ; letto
32E40F		LD (ADDL),A	; poni il byte in (ADDL)
17		RLA	; scambia tra di loro le due
17		RLA	; metà del byte
17		RLA	
17		RLA	
ED79		OUT (C),A	; uscita verso gli indicatori luminosi
DD23	DSM:	INC IX	; aggiorna il puntatore dello stack dei dati
DD23		IŃC IX	
DD23		INC IX	
00		NOP	; nessuna operazione
DD3600FF		LD (IX+00H),0FFH	; predisponi il tempo interno di DLOOPM
DD36010A		LD (IX+01H),00AH	; predisponi il tempo di CLOOPM
DD360202	CLOOPM:	LD (IX+02H),02H	; predisponi il tempo esterno di DLOOPM

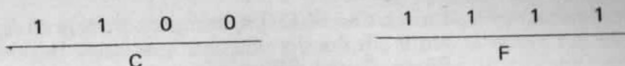
Codice oggetto		Codice sorgente	Commenti
21E50F		LD HL,ADDH	; punta al buffer del display
ED57		LD A,I	; cerca il valore di IFF2
EA4105		JP PE,HIGHM	
3600	LOWM:	LD (HL),00H	; valore = 0
1802		JR NEXTM	
3610	HIGHM:	LD (HL),10H	; valore = 1
ED73E20F	NEXTM:	LD (DATAL),SP	; trascrivi SP nel buffer DATAL
21B90F		LD HL,LEDL	; predisponi HL per CONVDI
11E50F		D DE,ADDH	; predisponi DE per CONVDI
CD7CFA		CALL CONVDI	
CD09F9	DLOOPM:	CALL DISPL	
DD3500		DEC (IX+00)	; temporizzatore per la visualizzazione
20F8		JR NZ,DLOOPM	
DD3502		DEC (IX+02)	; temporizzatore per la visualizzazione
20F3		JR NZ,DLOOPM	
DD3501		DEC (IX+01)	; temporizzatore per la routine di servizio
20CF		JR NZ,CLOOPM	
FDE1		POP IY	; ripristina i contenuti di ADDL
FD22E40F		LD (ADDL),IY	
FDE1		POP IY	; ripristina i registri della CPU
DDE1		POP IX	
F1		POP AF	
E1		POP HL	
D1		POP DE	
C1		POP BC	
FB		EI	; abilita le interruzioni
ED4D		RETI	; ritorno dall'interruzione

Passo 1

Il Modo 3 consente di definire con una notevole flessibilità le configurazioni e le proprietà delle porte del PIO. Occorre quindi selezionare e specificare al PIO, tramite dei byte di comando, numerose opzioni. Per vedere come si può programmare la porta del PIO per il funzionamento nel Modo 3, esaminiamo da vicino la routine INITPM.

INITPM: Si tratta di una routine di inizializzazione che, tra le sue altre funzioni, può programmare la Porta D del PIO per il funzionamento nel Modo 3. Di seguito vi elenchiamo tutte le funzioni delle routine INITPM, evidenziando in particolare quelle relative al funzionamento nel Modo 3. La routine effettua le seguenti operazioni:

a. Definisce per la CPU il Modo 2 di Interruzione.
b. Inizializza il registro I.
c. Carica l'indirizzo di SERVM nella Tabella dei Vettori di Interruzione.
d. Scrive il vettore di interruzione nella Porta D del PIO N. 2
e. Programma la Porta D del PIO per il funzionamento nel Modo 3: ciò viene attuato scrivendo il seguente byte di controllo nella Porta di controllo D, di indirizzo 0B:



La parte meno significativa indica che si tratta del byte di selezione del modo, mentre i bit D7 e D6 selezionano appunto il Modo 3.

f. Definisce il Registro di Selezione I/O. Se il byte di selezione del modo indica il Modo 3, il successivo byte di controllo viene interpretato come un byte che definisce il registro di selezione di I/O. Tale registro è formato da otto bit, il cui stato ha il significato seguente:

Se il bit Dn è a 1, la linea n (PDn) della porta del PIO viene considerata una linea di INPUT (ingresso).

Se il bit Dn è a 0, la linea n (PDn) della porta del PIO viene considerata una linea di OUTPUT (uscita).

La routine INITPM pone in uscita il byte

0 0 0 0 1 1 1 1

In questo modo vengono selezionate come uscite le linee PD7, PD6, PD5 e PD4, mentre le linee PD3, PD2, PD1 e PD0 vengono qualificate come ingressi.

g. Carica la parola di controllo dell'interruzione. Per il Modo 3 devono essere definiti i seguenti bit:

(a) *Bit di abilitazione all'interruzione:* le interruzioni vengono abilitate solo se questo bit è posto a 1. La routine INITPM lo pone a 1.

(b) *Bit AND/OR:* Se questo bit è posto a 1, tutte le linee dati non mascherate della porta devono essere attivate perchè il PIO generi una interruzione.

Se questo bit è posto a 0 è invece sufficiente che diventi attiva una sola delle linee dati non mascherate della porta perchè il PIO generi una richiesta di interruzione.

La routine INITPM pone a 0 questo bit.

(c) *Bit Alto/basso:* Questo bit specifica lo stato attivo delle linee della porta.
Se il bit Alto/Basso è 1, tutte le linee della porta saranno attive alte; se, invece, il bit è a 0, lo stato attivo sarà quello basso.

INITPM pone a 0 questo bit.

(d) *Bit Segue Maschera:* Se questo bit è posto a 1, il PIO viene programmato in modo da interpretare il successivo byte di comando scritto nella porta come parola di maschera (vedi altre).

La routine INITPM pone a 1 questo bit.

(e) I bit da D3 a D0 vengono posti a 0111 ad indicare che questo byte di controllo definisce la parola di controllo dell'interruzione.
Riassumendo, la parola di controllo dell'interruzione deve presentarsi nel formato seguente:

Abilitazione Interruzione	AND/ OR	Alto/ Basso	Byte segue Maschera	0	1	1	1

La routine INITPM definisce il byte di Controllo dell'Interruzione in modo che tutte le linee dati non mascherate della Porta D del PIO riconoscano la presenza di uno 0 logico. Non appena viene individuato uno zero, si ha la generazione dell'interruzione.

(h) *Definisce il byte di maschera:* il successivo byte di controllo posto in uscita dalla routine INITPM verso la Porta D è il byte di maschera, come richiesto dallo stato del bit D4 nella parola di controllo dell'interruzione. Questo byte definisce quali linee dati della porta sono mascherate e quali non lo sono. Ogni bit, posto a zero nel byte di maschera, corrisponde a una linea dati non mascherata.

INITPM pone in uscita il seguente byte di Maschera:

1 1 1 1 1 1 0 0

In questo modo la Porta D del PIO viene programmata in modo tale che viene controllata la presenza di uno 0 solo sulle linee PD0 e PD1.

SERVM: Questa routine di servizio dell'interruzione svolge le seguenti funzioni:

a. Salva lo stato della CPU.
b. Legge le quattro linee di ingresso della Porta D del PIO N. 2-B.
c. Visualizza sul miniterminale, per circa 5 secondi, la configurazione letta come un byte esadecimale (cifra più significativa posta a zero).
d. Pone in uscita la configurazione letta sulle quattro linee di uscita della Porta Dati D del PIO. Ciò richiede lo scambio della parte bassa con quella alta del byte.
e. Ripristina lo stato della CPU.
f. Abilita le interruzioni mascherabili della CPU.
g. Restituisce il controllo alla routine MAIN.

Nella Figura 7-19 è illustrato il flusso del controllo tra le routine INITPM, MAIN e SERVM.

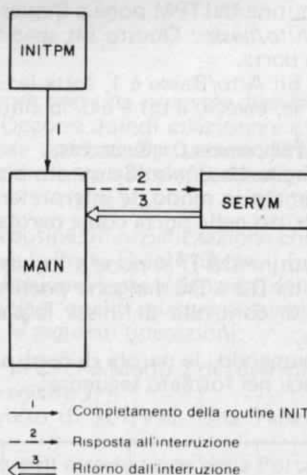


Figura 7-19. Flusso del controllo tra le routine INITPM, MAIN e SERVM.

Per quanto riguarda lo stato dei segnali di handshake durante il funzionamento del PIO nel Modo 3, occorre segnalare un altro fatto. Se la Porta C opera nel Modo 3, la linea CRDY viene mantenuta bassa, se, invece, è la Porta D a funzionare nel Modo 3, è la linea DRDY ad essere tenuta bassa. L'unica eccezione si verifica quando la Porta C opera nel Modo Bidirezionale; in tal caso per la Porta C sono necessarie anche le linee di handshake della Porta D. Ricordate infatti che nell'Esperimento N. 5, potevate usare il segnale DRDY, anche se la Porta F era programmata per il funzionamento nel Modo 3.

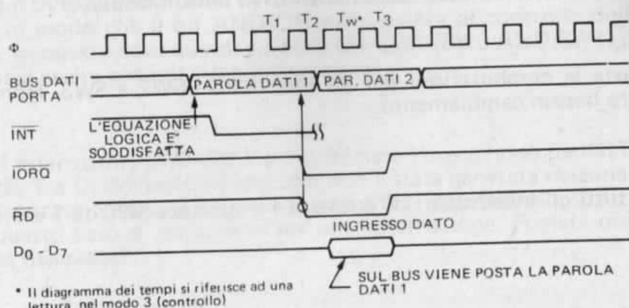


Figura 7-20. Diagramma dei tempi per il funzionamento del PIO nel modo 3.

Passo 2

Nella Figura 7-20 sono riportate le temporizzazioni nel caso che il PIO operi nel Modo 3. Come si nota, in tale modo la CPU può eseguire una normale operazione di lettura o scrittura sulla porta del PIO. La temporizzazione per un'operazione di scrittura è la stessa che si aveva per il Modo 0.
In particolare, nella Figura 7-20 è illustrata un'operazione di lettura. I dati inviati alla CPU saranno composti sia da dati "in uscita", provenienti dalle linee designate come linee di uscita, sia da dati in ingresso provenienti da linee di ingresso. Il contenuto dei bit, le cui linee corrispondenti sono state definite come uscite, rispecchia il contenuto dell'ultimo byte di dato posto in uscita verso la porta in Modo 3. Il dato presente sulle linee definite come ingressi dipende invece dallo stato degli ingressi dati della porta in Modo 3, immediatamente prima del fronte di discesa del segnale RD.

Il PIO controlla continuamente le linee dati non mascherate della porta verificando se si presenta la configurazione definita per generare una interruzione. Quando tale configurazione viene riconosciuta, viene generata una interruzione. Prima che ne venga generata un'altra è necessario che la condizione di generazione (la configurazione così definita per la porta) diventi prima falsa e poi di nuovo vera. Supponiamo, per esempio, che il PIO sia stato programmato per generare una interruzione se una qualsiasi linea di dati diventa attiva. Inoltre, supponiamo che una linea di dati diventi attiva e il PIO generi una interruzione. Se diventa attiva un'altra linea, diversa dalla precedente non viene generata nessuna interruzione; analogamente, se la linea di dati attiva iniziale diventa attiva contemporaneamente ad un'altra, verrebbe generata una sola interruzione. Prima che il PIO generi un'altra interruzione, tutte le linee di dati devono diventare non attive e poi almeno una di esse deve ritornare di nuovo attiva.

Passo 3

Collegate il circuito previsto per questo esperimento come indicato nella Figura 7-18. Ponete a 1 gli interruttori SW0-SW3. Iniziate l'esecuzione dalla locazione INITPM, quindi spostate l'interruttore SW3 da 1 a 0 logico. Cosa osservate?

Noi non abbiamo osservato alcun cambiamento in nessun display.

Passo 4

Provate tutte le combinazioni per gli interruttori SW2 e SW3. Non dovreste comunque notare nessun cambiamento.

Passo 5

Riportate tutti gli interruttori di nuovo a 1 e spostate SW0 da 1 a 0. Cosa notate?

Noi abbiamo notato che viene visualizzato un "E" esadecimale sui quattro indicatori luminosi LM4-LM7 e uno 0E sul display dell'unità tastiera/display del Nanocomputer. Noterete inoltre che la routine di servizio dell'interruzione restituisce il controllo alla routine MAIN anche se SW0 continua a rimanere a 0. E' stata generata solo una interruzione. Questo dimostra che, per il PIO nel Modo 3, una volta che le linee di dati hanno assunto uno stato che genera una interruzione, non si avranno altre interruzioni fintantochè lo stato logico delle linee non cambi, passando attraverso un'altra configurazione che non provoca la generazione di una interruzione.

Passo 6

Quando il display ritorna a decrementare il contatore, cioè l'interruzione precedentemente generata è stata servita e il controllo è stato restituito alla routine MAIN, spostate SW1 da 1 a 0. Cosa notate?

Noi non abbiamo notato alcun cambiamento, vale a dire non è stata generata nessuna interruzione.

Passo 7

Dovete infatti modificare gli interruttori in modo tale che la condizione di generazione dell'interruzione non continui ad essere presente.
Riportate quindi SW0 e SW1 a 1, e riportate poi SW1 di nuovo a 0. A questo punto si che dovreste notare che è stata generata una interruzione.

Passo 8

Con tutti e quattro gli interruttori a 1, spostate avanti e indietro SW0 appena appena, il più delicatamente possibile. Dopo qualche tentativo, forse riuscirete a generare una interruzione e riportare SW0 a 1 in un tempo abbastanza breve per permettere al

display di uscita di visualizzare tutti 1. Ciò dimostra quanto tempo occorre alla routine di servizio della interruzione SERVM per salvare lo stato della CPU, cioè circa 53 microsecondi. Questo è approssimativamente l'intervallo di tempo che intercorre tra la generazione dell'interruzione e l'attivazione del segnale RD da parte dell'istruzione IN. Avremo potuto memorizzare il segnale SW0 se fosse stato estremamente importante sapere quale linea aveva causato l'interruzione.

Passo 9

Modificate il byte alla locazione CWORD+1 della routine INITPM da 97 a D7. Così facendo fate in modo che il bit AND/OR nella parola di controllo dell'interruzione sia posto a 1. In questo caso quindi sia SW0 che SW1 (SW0 AND SW1) devono essere a 0 perchè possa essere generata una interruzione.

Passo 10

Portate gli interruttori SW0-SW3 a 1 e iniziate l'esecuzione da INITPM. Spostate quindi SW0 da 1 a 0: dovreste notare che non è stata generata nessuna interruzione. Dopo aver riportato SW0 a 1, riprovate con SW1, spostandolo su 0. Noterete che neanche in questo caso è stata generata una interruzione. Portate ora sia SW0 che SW1 a 0. Cosa osservate?

A questo punto sì che dovreste notare l'avvenuta generazione di una interruzione.

Passo 11

Non appena l'interruzione è stata servita, spostate SW0 da 0 a 1 e quindi di nuovo a 0. Cosa osservate?

Noi abbiamo osservato che viene generata un'altra interruzione. La condizione "AND" per cui sia SW0 che SW1 devono essere uguali a 0 perchè possa essere generata l'interruzione, è diventata falsa e poi vera di nuovo. Quindi l'interruzione può essere generata.

Notate che in questo esperimento le linee di handshake $\overline{STB}$ e RDY non hanno avuto alcun ruolo.

ESPERIMENTO N. 7

Scopo

Lo scopo di questo esperimento è quello di illustrare lo schema interno di priorità delle interruzioni del dispositivo PIO. Poichè entrambe lo porte del PIO sono in grado di generare interruzione è necessaria una logica di priorità tra le porte stesse, che risolva in modo prevedibile una eventuale contemporaneità delle richieste alla CPU.

Schema del circuito (Figura 7-21)

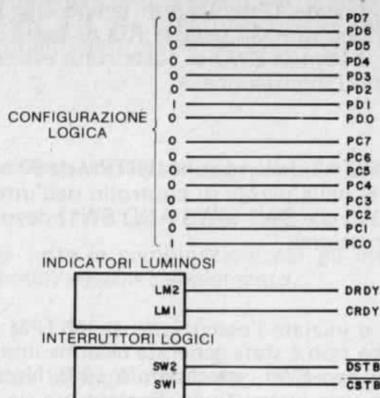


Figura 7-21. Schema N. 5.

Programmi: MAIN, SERVIC, SERVID e INITPP

Oltre alla routine MAIN, SERVIC e SERVID, in questo esperimento verrà usata la seguente routine:

Codice oggetto		Codice sorgente	Commenti
ED5E	INITPP:	IM2	; modo 2 di interruzione dello Z80
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; predisponi il vettore di interruzione
FD211904		LD IY,SERVIC	; indirizzo della routine di servizio della ; porta C
FD220A0F		LD (TABLE+0AH),IY	; inseriscilo nella tabella
FD211F04		LD IY,SERVID	; indirizzo routine di servizio della porta D
FD22080F		LD (TABLE+08H),IY	; inseriscilo nella tabella
3E0A		LD A,0AH	; predisponi il vettore di interruzione rela- ; tivo alla porta C
D30A		OUT (0AH),A	
3E08		LD A,08H	; predisponi il vettore di interruzione rela- ; tivo alla porta D
D30B		OUT (0BH),A	
08		EX AF,AF'	; predisponi il formato per CONVDI
3E40		LD A,40H	
08		EX AF,AF'	
3E4F		LD A,4FH	; modo 1 per C e D
D30A		OUT (0AH),A	
D30B		OUT (0BH),A	
3E87		LD A,87H	; abilita C e D
D30A		OUT (0AH),A	
D30B		OUT (0BH),A	
DB08		IN A,(08H)	; inizializza CRDY
DB09		IN A,(09H)	; e DRDY
C3C302		JP MAIN	

Passo 1

Collegate il circuito dell'esperimento secondo lo schema di Figura 7-21. Noterete che, a causa della necessità di collegare 16 linee dati, non sono stati usati gli interruttori, ma sono stati indicati solo gli "stati logici" delle linee stesse. Per implementare su una certa linea uno "stato logico" pari a 1 collegate la linea stessa alla tensione +5V, ai fori cioè dell'alimentazione presenti nella basetta, in una posizione qualsiasi. Allo stesso modo, per avere uno "stato logico" 0, collegate la linea ad un foro di GND (massa) sulla basetta.

Passo 2

Oltre alla routine MAIN, in questo esperimento utilizzeremo la routine INITPP e le due routine di servizio dell'interruzione SERVIC e SERVID. L'unica routine nuova è la routine INITPP, di cui parleremo di seguito. Negli esperimenti precedenti, la routine SERVID era stata usata per illustrare il funzionamento del PIO nel Modo 1, mentre la routine SERVIC era stata usata per illustrare il funzionamento del PIO nel Modo 2. Rivediamo le principali funzioni di queste routine.

INITPP: PP sta per Priorità del PIO. Questa routine di inizializzazione svolge le seguenti funzioni:

a. Definisce per la CPU il Modo 2 di Interruzione.
b. Inizializza il registro I.
c. Prepara la Tabella dei Vettori di Interruzione.
d. Carica nelle Porte C e D del PIO i rispettivi vettori di interruzione.
e. Programma le Porte C e D per il Modo 1.
f. Definisce la Parola di Controllo dell'Interruzione per entrambe le porte, abilitando le interruzioni.
g. Inizializza CRDY e DRDY.

SERVIC e SERVID: Queste due routine di servizio dell'interruzione svolgono le funzioni seguenti:

a. Salvano lo stato della CPU.
b. Leggono il byte proveniente rispettivamente dalle Porte C e D del PIO.
c. Visualizzano per 10 secondi sull'unità tastiera/display del Nanocomputer il byte di ingresso al posto del contatore.
d. Ripristinano lo stato della CPU.
e. Effettuano un ritorno dall'interruzione.

Dovreste ricordare che queste due routine condividono di fatto il segmento di codice SERVI.
Nella Figura 7-22 è schematicamente riportato il flusso del controllo tra le routine INITPP, MAIN, SERVIC e SERVID.

Apportate la seguente modifica al segmento di codice SERVI.

Sostituite l'istruzione NOP alla locazione ENABG con un'istruzione EI (FB esadecimale).

In questo modo avete inserito una modifica che abilita l'interruzione anche durante la maggior parte del tempo in cui viene eseguita ciascuna routine di servizio della interruzione. Ne consegue che una richiesta di interruzione che arrivi mentre è in corso il servizio di una interruzione precedente, può essere riconosciuta dalla CPU Z80.
La sospensione di un servizio di una interruzione in favore di un'altra viene definita con il termine di *prelazione* della prima interruzione. Le routine di servizio della interruzione che non riabilitano le interruzioni, impiegano quindi un metodo di servizio *senza prelazione*.

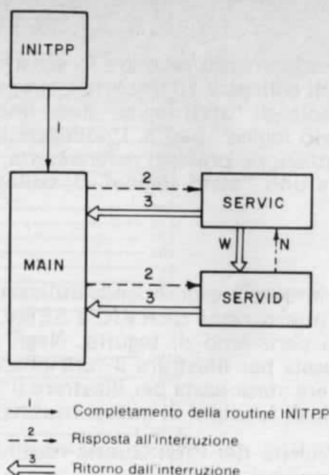


Figura 7-22. Flusso del controllo tra le routine INITPP, MAIN, SERVIC e SERVID.

Passo 3

Iniziate l'esecuzione da INITPP, e generate una interruzione muovendo su e giù l'interruttore SW1. Cosa notate?

Noi abbiamo notato che per circa 10 secondi compariva sulla tastiera/display del Nanocomputer il valore 01, presente sulla Porta C, mentre CRDY rimaneva, apparentemente, a 1.

Passo 4

Quando il puntatore di stack ritorna a 0F00, generate un segnale attivo $\overline{DSTB}$ muovendo l'interruttore SW2. Cosa osservate?

Noi abbiamo notato che sull'unità tastiera/display del Nanocomputer appariva per circa 10 secondi il valore presente sulla Porta D, cioè 02, mentre DRDY sembrava rimanere a 1.

Passo 5

Quando l'ultimo servizio dell'interruzione è stato completato, generate un segnale di strobe sulla Porta C con l'interruttore SW1. Poi, mentre è in esecuzione SERVIC, generate uno strobe per la Porta D tramite l'interruttore SW2. Che cosa notate?

Noi abbiamo notato che mentre inviavamo il segnale di strobe sulla Porta D, DRDY diventava bassa e il PIO memorizzava una interruzione pendente. Non appena la

routine SERVIC aveva completato l'esecuzione, il controllo della CPU veniva trasferito alla routine SERVID in risposta all'interruzione finora pendente della Porta D.

Passo 6

Quando il puntatore di stack contiene di nuovo 0F00, inviate un segnale di strobe sulla Porta D e, mentre la routine SERVID è ancora in fase di esecuzione, inviate un segnale di strobe sulla Porta C. Cosa notate?

In questo caso abbiamo notato un servizio delle interruzioni ANNIDATO (detto anche innestato, nidificato = nested). Quando abbiamo inviato il segnale di strobe sulla Porta C, abbiamo notato che il suo byte in ingresso, 01, veniva visualizzato sull'unità tastiera/display del Nanocomputer al posto di 02, il byte di ingresso della Porta D. La spiegazione di ciò sta nel fatto che la Porta C ha una priorità maggiore della Porta D. Quindi, il servizio della Porta D verrà interrotto a favore del servizio della Porta C, mentre l'inverso non potrà verificarsi. Di conseguenza, quando la Porta D chiede una interruzione mentre è in corso il servizio della Porta C la Porta D viene tenuta in attesa (pendente) fino a che non è stato completato il servizio della Porta C. Viceversa, una richiesta di interruzione fatta dalla Porta C causerà l'immediata sospensione del servizio della Porta D fino a quando il servizio della Porta C non sarà stato completato. Questa relazione tra C e D, in cui C ha una priorità maggiore di D, può anche essere espressa nel modo seguente:

$$C > D$$

Passo 7

Con il puntatore di stack su 0F00, inviate un segnale di strobe alla Porta C. Mentre la routine SERVID è in corso di esecuzione, inviate un ulteriore segnale di strobe sulla Porta D. Cosa notate?

La seconda interruzione viene tenuta pendente in attesa del completamento della prima. Ne consegue che:

$$D \ngtr D$$

Si spiega così la funzione del flip-flop di Interruzione Sotto Servizio interno al PIO. Il circuito interno al PIO, che contiene i flip-flop di Interruzione Pendente e di Interruzione Sotto Servizio, è illustrato nella Figura 7-23. Tale logica di controllo dell'interruzione è presente anche nel dispositivo CTC Z80 (circuito contaeventi-contatempi) e verrà così discusso più approfonditamente nel Capitolo 9.

ESPERIMENTO N. 8

Scopo

Lo scopo di questo esperimento è quello di illustrare la catena di priorità delle interruzioni chiamata in inglese *"daisy chain"* (lett.: ghirlanda) e realizzata nel PIO e negli altri dispositivi di supporto tramite le linee IEI e IEO.
Così, ad esempio, il PIO N. 1 e il PIO N. 2 presenti sulla scheda del Nanocomputer, sono configurati in daisy chain. In questo esperimento, aggiungeremo a questa catena un terzo PIO.

Figura 7-23. Logica di controllo dell'interruzione del PIO.

Schema del circuito (Figura 7-24)

Oltre al circuito presentato in Figura 7-21 in questo esperimento utilizzeremo anche il circuito della Figura 7-24.

Programmi

In questo esperimento useremo le seguenti routine: INITDC, INITPP, MAIN, SERVIC, SERVID, SERVIE e SERVIF. Negli esperimenti precedenti nel Capitolo 6 e di questo capitolo abbiamo già visto sia i listing che la documentazione relativa alle routine INITPP, MAIN, SERVIC e SERVID.
Come potete notare, esiste una notevole somiglianza tra le routine SERVIE e SERVIF e le routine SERVIC e SERVID.
Queste quattro routine differiscono solo per gli indirizzi delle porte, indicate simbolicamente con PORTC, PORTD, PORTE e PORTF. E' evidente che non è conveniente tenere quattro subroutine di queste dimensioni che differiscono solo per un byte. Noi le abbiamo utilizzate tutte e quattro per motivi di chiarezza, non certo per efficienza.

Codice oggetto		Codice sorgente	Commenti
ED5E	INITDC:	IM2	; modo 2 di interruzione dello Z80
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; predisponi il vettore di interruzione
FD212504		LD IY,SERVIE	; indirizzo routine di servizio per ingresso
FD220E0F		LD (TABLE+0EH),IY	; dalla porta E
			; inseriscilo nella tabella
FD212B04		LD IY,SERVIF	; indirizzo routine di servizio per
FD22100F		LD (TABLE+10H),IY	; ingresso dalla porta F
			; inseriscilo nella tabella
3E0E		LD A,0EH	; carica il vettore di interruzione E
D30E		OUT (0EH),A	
3E10		LD A,10H	; carica il vettore di interruzione F
D30F		OUT (0FH),A	
08		EX AF,AF'	; predisponi il formato per CONVDI
3E40		LD A,40H	
08		EX AF,AF'	
3E4F		LD A,4FH	; predisponi il modo 1 del PIO
D30E		OUT (0EH),A	; porta E
D30F		OUT (0FH),A	; porta F
3E87		LD A,87H	; abilita il PIO
D30E		OUT (0EH),A	; porta E
D30F		OUT (0FH),A	; porta F
DB0C		IN A,(0CH)	; inizializza ERDY
DB0D		IN A,(0DH)	; inizializza FRDY
C37305		JP INITPP	

Codice oggetto		Codice sorgente	Commenti
C5	SERVIE:	PUSH BC	
0E0C		LD C,0CH	; interruzione della porta E
C33104	JP SERVI		

Codice oggetto		Codice sorgente	Commenti
C5	SERVIF:	PUSH BC	
0E0D		LD C,0DH	; interruzione della porta F
C33104		JP SERVI	

Passo 1

In questo esperimento faremo largo uso delle porte A e B del PIO N. 2 (indicate con C e D sulla scheda del Nanocomputer), e delle Porte A e B del PIO N. 3, un nuovo PIO che dovrete inserire e collegare sulla basetta per collegamenti senza saldature del Nanocomputer. Nell'ultimo esperimento avrete notate che, nel circuito del PIO, la Porta A ha una priorità più elevata della Porta B. Qual'è la struttura delle priorità tra diversi dispositivi PIO? Qualsiasi struttura di priorità deve risolvere i "conflitti", le "concorrenze" tra le possibili richieste contemporanee di servizio. In questo esperimento cercheremo di chiarirci le idee su questo argomento.

Figura 7-24. Schema N. 6. Collegamenti per un PIO esterno.

Collegate il circuito come illustrato nello Schema N. 6 della Figura 7-24. Notate che l'ingresso $\overline{CE}$, piedino 4 del PIO, è collegato al segnale $\overline{IOQ3}$. Tale segnale decodifica le 6 linee A7-A2 del bus degli indirizzi ed è attivo se, e solo se, queste linee si trovano tutte allo stato logico seguente:

A7	A6	A5	A4	A3	A2
0	0	0	0	1	1

Poichè le linee A1 e A0 sono collegate a C/D e B/A, rispettivamente, tutte le linee di indirizzo sono decodificate in modo da ottenere i seguenti indirizzi di porta per il PIO N. 3:

Porta	Indirizzo
Porta Dati A	0C
Porta Dati B	0D
Porta di controllo A	0E
Porta di controllo B	0F

Nella Figura 7-25 è illustrato il circuito che collega il piedino $\overline{INT}$ del PIO al piedino $\overline{INT}$ della CPU Z80. I dispositivi open-collector (a collettore aperto) sono stati usati per la presenza sulla scheda del Nanocomputer di una resistenza di richiamo da 910

ohm. Per non invertire lo stato logico del segnale che viene inviato alla CPU sono stati necessari due invertitori. Tutte le resistenze sono resistenze di richiamo alla tensione positiva (pull-up).

Nella Figura 7-26 è illustrato il modo in cui i piedini IEI e IEO sono collegati su tutti e tre i PIO. Notate che il piedino IEI sul PIO N. 1 è collegato direttamente a +5V. Il piedino IEO sul PIO N. 1 è collegato alla linea IEI del PIO N. 2 mentre il piedino IEO sul PIO N. 2 è collegato alla linea IEI del PIO N. 3. Nessun collegamento invece viene effettuato al pin IEO del PIO N. 3.

Questo tipo di collegamenti è quello che si chiama *Daisy chain.*

I collegamenti a daisy chain generano una struttura, una catena di priorità, per le sei porte del PIO imponendo le seguenti regole:

- Il PIO può richiedere una interruzione se e solo se la propria linea d'ingresso IEI è alta.
- Se un PIO richiede una interruzione, pone la sua linea di uscita IEO bassa.
- Se il PIO "sente" che la linea IEI è bassa, pone bassa anche la linea IEO.

Tali norme comportano alcune conseguenze:

1. Il PIO N. 1 può richiedere una interruzione in qualunque momento, dato che la sua linea di ingresso IEI è sempre alta.
2. Se il PIO N. 1 richiede una interruzione, la sua linea IEO viene posta bassa. Il livello basso si propaga lungo la catena dei dispositivi poichè vengono via via posti bassi gli IEI del PIO N. 2 e del PIO N. 3.

   Più in generale, una richiesta di interruzione, proveniente da un PIO collegato all'interno di una daisy chain, fa sì che la linea di ingresso IEI diventi bassa per tutti i circuiti PIO "più bassi" presenti nella daisy chain.
3. Per quanto riguarda le sei porte del PIO, viene perciò imposta la seguente struttura di priorità:

   (PIO N. 1, A) > (PIO N. 1, B) > (PIO N. 2, A) > (PIO N. 2, B) > (PIO N. 3, A) > (PIO N. 3, B)

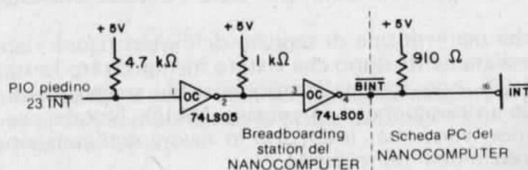


Figura 7-25. Collegamento di un circuito PIO esterno alla linea $\overline{INT}$ della CPU Z80.

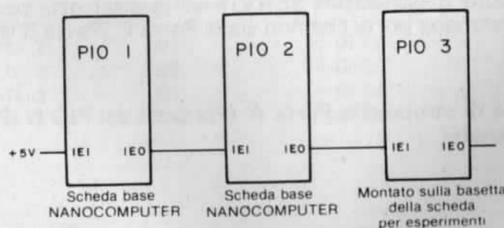


Figura 7-26. Daisy chain del PIO.

Nei passi che seguono verificheremo sperimentalmente queste affermazioni.

Passo 2

Come abbiamo detto in precedenza, useremo quattro porte, che in questo esperimento saranno quelle del PIO N. 2 e del PIO N. 3. Le routine INITDC e INITPP programmano tutte quattro le porte ad operare nel Modo 1, cioè nel Modo di Ingresso. Notate che la routine INITDC esegue per il PIO N. 3 gli stessi compiti che la routine INITPP effettua per il PIO N. 2.
Le quattro porte dati sono state collegate per avere in ingresso le seguenti configurazioni:

Porta	Byte in ingresso
Porta Dati A del PIO N. 2 (Porta C)	01
Porta Dati B del PIO N. 2 (Porta D)	02
Porta Dati A del PIO N. 3 (Porta E)	03
Porta Dati B del PIO N. 3 (Porta F)	04

Le quattro routine di servizio interruzione SERVIC, SERVID, SERVIE e SERVIF svolgono le stesse funzioni per le diverse porte. La più importante funzione, o meglio, la più evidente è quella di leggere e visualizzare il byte posto in ingresso della porta. Tra le porte del PIO e le routine di servizio dell'interruzione esiste la seguente corrispondenza:

Porta	Routine di servizio interruzione
Porta Dati A del PIO N. 2	SERVIC
Porta Dati B del PIO N. 2	SERVID
Porta Dati A del PIO N. 3	SERVIE
Porta Dati B del PIO N. 3	SERVIF

Verificate ora il contenuto della locazione: DSG +6 (cioè ENABG) e controllate che sia FB.
Siete così sicuri che ogni routine di servizio dell'interruzione riabilita le interruzioni mascherabili immediatamente dopo che è stato memorizzato lo stato della CPU.

Nella Figura 7-27 sono schematizzati i possibili trasferimenti di controllo tra le routine di servizio interruzione e la routine MAIN. Notate che l'esecuzione della routine SERVIC non verrà mai interrotta in favore dell'esecuzione di una qualsiasi altra routine di servizio dell'interruzione.
Allo stesso modo, l'esecuzione della routine SERVID potrà essere interrotta solo per eseguire la routine SERVIC, ma non lo sarà per eseguire né la routine SERVIE né la routine SERVIF.
Per contro, l'esecuzione della routine SERVIF verrà interrotta per servire le interruzioni generate da qualunque porta che non sia la Porta F (Porta B del PIO N. 3).

Passo 3

Inviate un segnale di strobe alla Porta A (Porta C) del PIO N. 2 premendo l'interruttore SW1. Cosa notate?

Noi abbiamo notato che sull'unità tastiera/display del Nanocomputer veniva visualizzato per circa 10 secondi il byte 01, mentre il puntatore di stack veniva decremen-

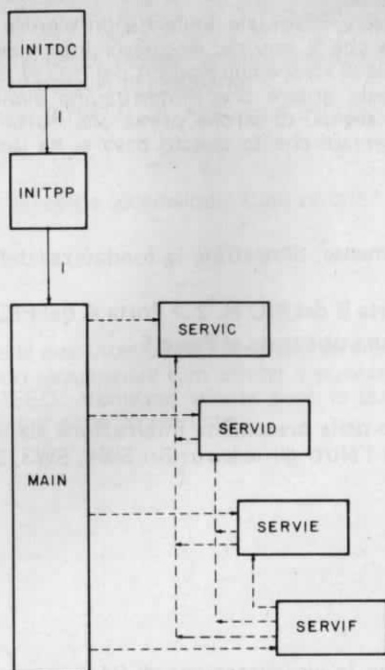


Figura 7-27. Flusso di controllo parziale tra le routine INITDC, INITPP, MAIN, SERVIC, SERVID, SERVIE e SERVIF.

tato a 0EEC e il flip-flop di interruzione IFF2 e di conseguenza IFF1 si portavano a 1.

Passo 4

Ripetete il passo per ciascuna porta del PIO, lasciando eseguire la routine di servizio dell'interruzione fino al suo completamento prima di provare con la porta successiva. Verificate quindi la seguente tabella:

Porta	Byte visualizzato	Contenuto del puntatore di stack	IFF1/IFF2
A del PIO N. 2	01	0EEC	1
B del PIO N. 2	02	0EEC	1
A del PIO N. 3	03	0EEC	1
B del PIO N. 3	04	0EEC	1

Passo 5

Verificate sperimentalmente che la Porta A del PIO N. 3 ha una priorità maggiore della Porta B dello stesso PIO.

Questa verifica può essere effettuata inviando un segnale di strobe alla Porta B del PIO N. 3 e poi, prima che il servizio di questa interruzione sia stato completato, inviando un altro segnale di strobe alla Porta A del PIO N. 3.
A questo punto dovreste notare che l'interruzione viene servita in modo ANNIDATO. Inviate quindi segnali di strobe prima alla Porta A e poi alla Porta B del PIO N. 3: dovreste osservare che in questo caso si ha un servizio dell'interruzione SEQUENZIALE.

Passo 6

Sempre sperimentalmente, dimostrate la fondatezza della nostra precedente affermazione:

Porta B del PIO N. 2 > Porta A del PIO N. 3

usando la stessa procedura utilizzata al Passo 5.

Passo 7

Prima che il servizio della precedente interruzione sia stato completato, premete rapidamente uno dopo l'altro gli interruttori SW4, SW3, SW2 e SW1. Cosa notate?

Noi abbiamo notato che la visualizzazione di 04 si interrompeva per permettere la visualizzazione di 03, associato ad una porta con priorità più elevata, il quale, a sua volta, veniva interrotto per visualizzare 01 e che infine era sospeso da 01. Una volta che il byte 01 era stato visualizzato per circa 10 secondi, veniva ripresa la visualizzazione di 02, quindi di 03 e infine di 04. Ecco una tabella che sintetizza un po' quanto siamo venuti dicendo:

Display	Contenuto del puntatore di stack
04	0EEC
03	0ED8
02	0EC4
01	0EB0
02	0EC4
03	0ED8
04	0EEC

Passo 8

Premete in ordine SW1, SW2, SW3 e SW4 in rapida successione e verificate ancora l'esattezza della tabella che segue che sintetizza quanto detto finora:

Display	Contenuto del registro del puntatore di stack
01	0EEC
02	0EEC
03	0EEC
04	0EEC

Passo 9

Provate con diverse sequenze di strobe, secondo l'ordine che volete, in modo da convincervi della validità della seguente relazione:

(Porta A PIO N. 2) > (Porta B PIO N. 2) > (Porta A PIO N. 3) > (Porta B PIO N. 3)

Passo 10

Premete SW1 due volte in rapida successione. Cosa notate?

Noi abbiamo osservato che le due interruzioni venivano servite una dopo l'altra. Il servizio avveniva in modo sequenziale e in effetti il puntatore di stack non veniva mai decrementato oltre 0EEC. Venivano servite però le due interruzioni perchè il byte 01 veniva visualizzato per circa 20 secondi.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]

[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]

CAPITOLO 8

UN TESTER PER CIRCUITI INTEGRATI TTL

INTRODUZIONE

Nel corso di questo capitolo ci proponiamo di presentare una dettagliata documentazione su un tester per circuiti integrati TTL in grado di provare un gran numero di sistemi a logica combinatoria a 14 e 16 pin. Il componente principale del tester è il PIO N. 2 della scheda del Nanocomputer. Il PIO è programmato per funzionare in Modo 3 (Control Mode) con *entrambe* le porte di I/O. Ne consegue che ognuna delle sue 16 linee di I/O può essere programmata, da software, come linea di ingresso oppure di uscita. Il tester, nella sua struttura base, è realizzato collegando i pin dei chip a 14 o 16 pin da provare alle due porte di I/O del PIO. La CPU, attraverso il PIO, imposta delle particolari configurazioni sugli ingressi del chip in prova e verifica, in corrispondenza di ogni configurazione, che le uscite di quest'ultimo si presentino corrette. All'efficienza del tester contribuisce in misura determinante la programmabilità del chip PIO, che consente di costruire un tester più semplice degli altri trattati dalla lettura corrente.

OBIETTIVI

Al termine di questo capitolo, dovreste essere in grado di:

- Servirvi di un tester per circuiti integrati per la verifica della funzionalità di diversi IC 74LS.
- Conoscere la relazione tra le posizioni dei bit sulle porte di I/O del PIO e la configurazione dei pin di un circuito integrato a 14 o a 16 pin.
- Comprendere il criterio seguito dalla routine CHPTST per selezionare il numero minimo di parole di prova tra le 65.536 possibili parole di prova per un circuito integrato a 14 oppure a 16 pin.
- Conoscere chiaramente le modalità secondo le quali l'azione congiunta dello hardware e del software del tester consente di determinare se un circuito integrato sia difettoso oppure no.

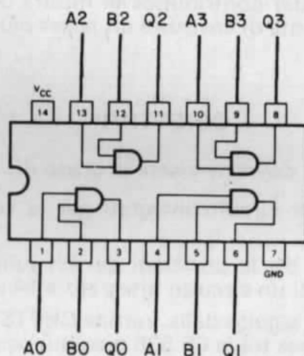
CRITERI OPERATIVI

Nel corso di questo capitolo ci proponiamo di fornire una descrizione dei criteri operativi, del software e dell'hardware necessari per un tester di circuiti integrati TTL low-power Schottky. Considerando il costo davvero minimo degli IC 74LS ci si potrebbe domandare perchè mai ci si voglia dedicare alla realizzazione di un tester del genere, con dispendio di tempo e vincolando un microcomputer all'intero procedimento. La risposta, nel nostro caso, è molto semplice. La costruzione di questo tester aveva lo scopo di eliminare i circuiti integrati difettosi da un migliaio

di IC utilizzati durante un corso di elettronica digitale proposto da 200 studenti durante un anno accademico al Virginia Polytechnic Institute and State University di Blacksburg, Virginia. Secondo la nostra esperienza, l'eliminazione degli IC difettosi consente di procedere più velocemente durante gli esperimenti di elettronica digitale.

La tecnica che utilizzeremo trova origine in due articoli pubblicati in riviste di hobbystica nel corso del 1978, un articolo di Mark Thorson del numero di giugno di *Byte* ed un articolo di continuazione di Joe Swank nel numero di settembre del *Dr. Dobbs Journal.* L'articolo di Thorson contiene la descrizione di un tester per circuiti integrati realizzato mediante componenti discreti e costituiti da sei memorie 256 × 1 74LS200, due contatori binari 74161 nonchè numerosi gate e flip-flop. Il chip tester di Thorson permette il test di un circuito integrato con un massimo di otto ingressi e sei uscite. Per la definizione delle caratteristiche del componente da provare con il tester (ingressi, uscite, pin non usati, alimentazioni) si usano dei porticelli. L'articolo di Swank riguarda l'impiego di un microcomputer KIM che migliora l'efficienza del tester di Thorson con la trasformazione di parte della logica hardware in software. Nonostante ciò, Swank si affida a ponticelli per segnalare le caratteristiche dei componenti in prova all'interfaccia del tester KIM. Nel corso di questo capitolo giungeremo ad esporre come l'utilizzazione del circuito PIO può condurre all'eliminazione di questi ponticelli. Il circuito del tester così realizzato risulta estremamente semplice, essendo basato principalmente su di un PIO ed una CPU Z80.

I principi fondamentali che stanno alla base del tester per circuiti integrati possono essere riassunti nel modo che segue. Il circuito integrato in prova è visto come un sistema digitale a ingressi/uscite multiple, che utilizza una logica-combinatoria e



(A) Schema di collegamento

B	A	Q
0	0	0
0	1	0
1	0	0
1	1	1

(B) Tabella della verità

Figura 8-1. Un quad AND gate a 2 ingressi 74LS08.

presenta una configurazione dei pin nota in anticipo. Per *logica–combinatoria* intendiamo che il chip "combina" vari stati logici statici presenti ai suoi ingressi per generare, in corrispondenza, dei valori logici statici sui suoi pin di uscita. Esempi elementari di logica–combinatoria sono rappresentati dalle porte logiche: AND, NAND, OR e NOR. Il termine combinatorio è giustificato dall'opportunità di operare una distinzione tra queste funzioni logiche fondamentalmente statiche e le proprietà più dinamiche dei sistemi logici ***sequenziali*** come i flip-flop ed i contatori. Per i dispositivi sequenziali sono, invece, le ***transizioni*** da un livello logico all'altro sulle uscite gli effetti della presenza di certi stati logici su alcuni pin di ingresso. In seguito si dedicherà un po' di spazio ad alcune migliorie che, apportate al tester qui presentato, potrebbero aprire il suo campo di applicazione a componenti della classe dei sistemi logici sequenziali.

Il procedimento di prova, seguito dal tester per sistemi logici combinatori illustrato in questo capitolo, consiste di tre fasi:

1. Generare la tabella della verità completa di un circuito integrato di riferimento che funziona correttamente.
2. Generare la tabella della verità completa relativa al circuito integrato "testato" o "sconosciuto".
3. Mettere a confronto le due tabelle della verità. L'assenza di coincidenza in qualche punto implica un qualche difetto nel circuito di prova.

A titolo di esempio si prenda in considerazione il 74LS08 (quattro porte AND a due ingressi) illustrato in Figura 8–1. Questo chip può essere considerato come la combinazione di quattro porte AND indipendenti a 2 ingressi, ciascuna delle quali è definita da una semplice tabella della verità con 2 ingressi e 1 uscita. Esisterebbero dunque quattro tabelle della verità distinte, ciascuna costituita da quattro soli elementi.

In alternativa, potreste considerare questo IC come un sistema logico combinatorio con otto ingressi e quattro uscite, come illustrato dalla Figura 8–2. Associato a

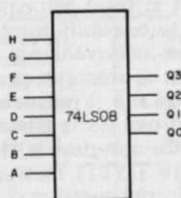


Figura 8–2. Rappresentazione alternativa di un 74LS08.

quest'ultimo sistema vi sarebbe una tabella della verità con 8 ingressi e 4 uscite, con un totale di 256 combinazioni possibili. La Tabella 8–1 riproduce una parte di questa tabella della verità. Come potrete vedere, questo modo più globale di considerare un chip vi permette di caratterizzarne le proprietà in una forma meglio adeguata per un tester.

I componenti funzionali occorrenti alla prova del "sistema" 74LS08, illustrato in Figura 8–2, sono presentati in Figura 8–3. Per mezzo di un contatore a 8 bit, all'ingresso del chip 74LS08 si applicano 256 differenti combinazioni di 8 bit. Le uscite di 4 bit sono memorizzate in 256 locazioni consecutive di una memoria a lettura/scrittura. Terminata la memorizzazione dell'intera tabella relativa alla risposta del chip in prova, il tester può procedere al confronto di questa tabella con quella di un circuito "noto come buono".

Si osservi come, indipendentemente dai limiti di questa particolare realizzazione, il tipo di tester per TTL illustrato in Figura 8–3 possa essere applicato a tutti i chip che presentino una logica combinatoria statica. Esempi di sistemi logici combinatori

Tabella 8-1. Tabella parziale della verità con 8 ingressi, 4 uscite.

Ingressi								Uscite			
H	G	F	E	D	C	B	A	Q0	Q1	Q2	Q3
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	1	0	1	1	1	0	0	0
0	0	0	0	1	1	0	0	0	1	0	0
0	0	0	0	1	1	0	1	0	1	0	0
0	0	0	0	1	1	1	0	0	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
				•					•		
				•					•		
				•					•		
				•					•		
1	1	1	1	1	1	1	1	1	1	1	1

sono rappresentati da porte, buffer, decodificatori, multiplexer, sommatori e dispositivi del chip non-edge triggered (ossia, che non cambiano stato in corrispondenza del fronte di salita o di discesa). Il criterio principale di questa strategia di prova è che il chip riceve in ingresso tutte le possibili combinazioni di stati logici statici. E' bene a questo punto accennare ad un inconveniente insito in questo metodo di prova. La sequenza degli ingressi è sempre la *stessa,* e precisamente quella ottenuta incrementando un contatore a *n* bit, dove *n* è il numero degli ingressi del chip. Anche se l'applicazione di questa tecnica di prova lascia passare inosservati soltanto i problemi più sottili ed intricati, la loro esistenza non deve tuttavia restare ignota.

Thorson ha realizzato in hardware TUTTI i componenti funzionali illustrati nella Figura 8-3. Scegliendo un approccio del tutto opposto, il tester descritto in questo capitolo elimina i ponticelli del circuito hardware e la logica di conteggio. Ne deriva che, di tutta la realizzazione hardware, rimangono soltanto la memoria, la CPU, quattro porte di I/O per il chip ed il sistema dei collegamenti fisici con il chip in

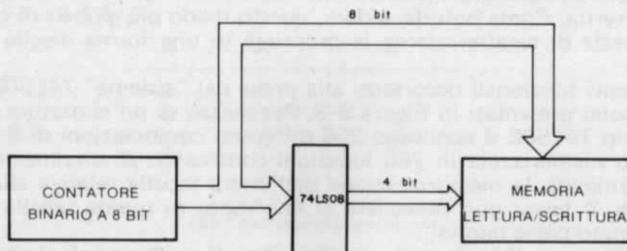


Figura 8-3. Componenti funzionali di un tester con circuiti integrati.

prova. I ponticelli sono sostituiti da due byte di maschera, uno per ciascuna delle due porte del PIO che funzionano in Control Mode. Il contatore è sostituito con del software che invia continuamente al IC in prova bytes ogni volta incrementati di una unità. Riguardo al componente da provare, è necessario conoscere quanto segue, per poter predisporre per la prova il tester:

1. La posizione dei pin di alimentazione: essi devono essere collegati alle relative linee di alimentazione.
2. La configurazione dei pin; deve cioè, essere nota la posizione di tutti i pin di ingresso e di uscita.
3. Devono essere noti i pin per i quali non sono previsti collegamenti (NC), che devono essere collegati o a +5V o a massa.
4. Il chip non deve consistere in un dispositivo analogico-digitale ibrido, il cui funzionamento dipende da circuiti di temporizzazione interni o esterni. Inoltre il chip non deve presentare ingressi edge-activated (attivati dal fronte del segnale).

Esempi del tipo 3 sono offerti dai chip 74LS20 e 64LS30.
Esempi del tipo 4 sono offerti dai chip di multivibratori monostabili 74121, 74122 e 74123 e dai flip-flop 74LS74.

Nella Tabella 8-2 sono riportate le caratteristiche di ingresso/uscita di alcuni sistemi logici combinatori appartenenti alla serie TTL 74LS00. Ogni circuito integrato elencato presenta 14 oppure 16 pin e può essere provato valendosi del tester descritto in questo capitolo. Il software, che descriveremo nei particolari più avanti, necessita di 4 K byte di memoria a lettura/scrittura per poter provare componenti con al massimo nove pin di ingresso. Per componenti con un numero di pin di ingresso maggiore

Tabella 8-2. Caratteristiche di ingresso/uscita di alcuni sistemi logici combinatori appartenenti alla serie 74LS.

Circuito integrato	Numero di bit "di uscita" (in uscita dal microcomputer, in ingresso all'integrato)	Numero di bit "di ingresso" (in ingresso al microcomputer, in uscita dall'integrato)	Numero di pin "nessun collegamento"
74LS02	8	4	0
74LS04	6	6	0
74LS05	6	6	0
74LS08	8	4	0
74LS30	8	1	3
74LS32	8	4	0
74LS42	4	10	0
74LS125	8	4	0
74LS139	6	8	0
74LS365	8	6	0
74LS54	10	1	1

si procede al collegamento alternato a massa e a +5V dei pin eccedenti provando i chip in più tempi sino ad esaurimento di tutte le possibili configurazioni degli ingressi.

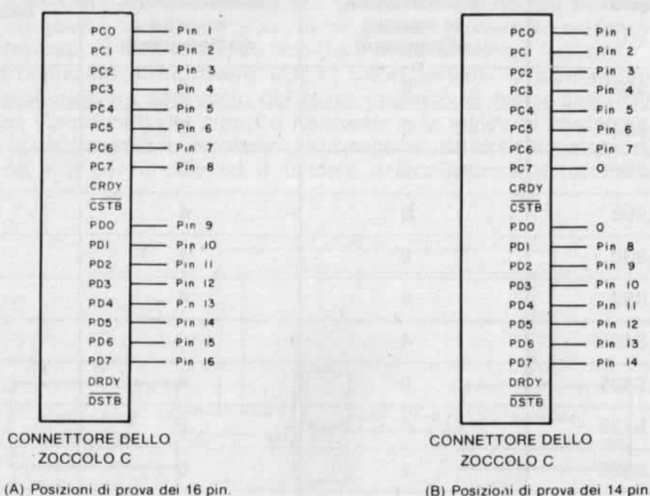
Per ampliare la capacità del tester, in modo da potervi alloggiare sistemi logici più grandi, vi sono diverse possibilità:

1. Aggiungere delle memorie addizionali.
2. Riscrivere il software in modo da impostare le configurazioni dei bit in ingresso e confrontare le uscite che ne risultano, per blocchi.
3. Riscrivere il software in modo da confrontare byte-per-byte le uscite del chip di riferimento (supporto "buono") e quelle dei componenti sconosciuti.

L'ultima alternativa è quella che permette il maggiore risparmio di spazio di memoria, ma necessita di hardware addizionale, e in particolare di un secondo chip PIO. Poichè il costo della memoria addizionale è superiore a quello di un circuito PIO addizionale (al momento attuale il suo prezzo è di circa $ 25), la terza alternativa è quasi certamente quella da preferirsi.

CIRCUITO DI INTERFACCIA

I circuiti di interfaccia riprodotti nelle Figure 8-4A, 8-4B, tutto sommato piuttosto semplici, costituiscono l'interfaccia con il tester del Nanocomputer.
Il PIO N. 2, sulla scheda del Nanocomputer, è interfacciato con il componente in prova attraverso gli zoccoli, sui quali sono presenti i segnali del PIO, situati sulla breadboarding station del Nanocomputer. Le due porte del PIO, denominate Porta C e Porta D, sono collegate per mezzo di fili, alle 16 posizioni dei pin di un DIP standard a 16 pin che serve per definire la posizione di prova sia per il circuito integrato di riferimento che per quello da provare. Un componente a 16 pin occupa interamente la posizione di prova, mentre un circuito integrato a 14 pin va collocato sui pin da 1 a 7 e da 10 a 16, mettendo a massa i pin inutilizzati 8 e 9. Le Figure 8-4A e B illustrano la corrispondenza tra le lineedelle porte dei dati del PIO, PC0-PC7



(A) Posizioni di prova dei 16 pin.

(B) Posizioni di prova dei 14 pin.

Figura 8-4. Schema del circuito del tester.

e PD0–PD7, ed i numeri dei pin degli integrati da provare a 16 e 14 pin rispettivamente. Notate che in Figura 8-4B le linee PC7 e PD0 non sono collegate a nessuno dei pin del componente a 14 pin, da provare. In quanto inutilizzate, queste linee devono essere poste a massa.

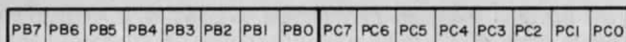
Poichè, come abbiamo già accennato, sia la Porta C che quella D funzionano nel Modo 3, ciascuna delle 16 linee delle porte di I/O è programmata per essere un ingresso o un'uscita mediante un byte di controllo della configurazione di I/O. Ricordiamo qui il significato dello stato dei bit del byte di controllo della configurazione di I/O del PIO:

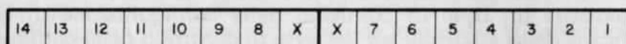
Bit Dn uguale a 0 nel byte di controllo di I/O della Porta C implica che la linea PCn è una linea di uscita.
Bit Dn uguale a 1 nel byte di controllo di I/O della Porta C implica che la linea PCn è una linea di ingresso.
Bit Dn uguale a 0 nel byte di controllo di I/O della Porta D implica che la linea PDn è una linea di uscita.
Bit Dn uguale a 1 nel byte di controllo di I/O della Porta D implica che la linea PDn è una linea di ingresso.

Il byte di controllo della configurazione di I/O o byte di maschera costituisce un parametro specificato dall'utente al tester, per tenere conto delle caratteristiche fondamentali del componente da provare. Per definire il byte di maschera associato al chip da provare si utilizza la procedura seguente:

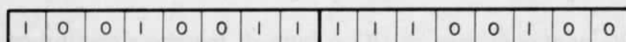
1. I pin di alimentazione del chip, +5V e massa, sono considerati ingressi del microcomputer, per cui alle linee delle porte del PIO ad essi associati è assegnato il valore logico 1 nel byte di controllo di I/O. Si badi che i pin di alimentazione devono essere collegati in modo corretto a +5V oppure a massa.
2. I pin degli ingressi del chip da provare sono considerati uscite del microcomputer, per cui alle linee delle porte del PIO ad essi associati è assegnato il valore logico 0 nel byte di controllo della configurazione di I/O.
3. I pin delle uscite del chip da provare sono considerati ingressi del microcomputer, per cui si trovano assegnato il valore 1 nelle posizioni dei bit ad essi relativi del byte di controllo.
4. I pin "nessun collegamento" del chip da provare devono essere posti a massa e sono considerati ingressi del microcomputer. I bit ad essi associati nel byte di controllo di I/O sono perciò posti al valore logico 1.
5. Per la prova di un chip a 14 pin i bit PC7 e PD0 sono considerati ingressi, per cui ad esse è assegnato il valore logico 1 nei loro rispettivi byte di controllo di I/O.

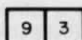
Avendo stabilito come debbono essere settati i vari bit dei due byte di controllo della configurazione di I/O, l'utente passa queste informazioni al tester valendosi di due locazioni di memoria, aventi etichetta MASKW e MASKW+1. L'utente provvede al caricamento manuale della locazione MASKW con il byte di controllo della configurazione di I/O relativo alla Porta C e della locazione MASKW+1 con quello relativo alla Porta D. Riassumendo, una linea della porta dei dati del PIO corrispondente ad un pin di uscita del chip da provare è utilizzata come INGRESSO AL MICROCOMPUTER, con assegnazione del valore logico 1. Analogamente, una linea della porta dei dati del PIO corrispondente ad un pin di ingresso del chip da provare è utilizzata come USCITA DAL MICROCOMPUTER, con assegnazione del valore logico 0. I pin di alimentazione e quelli "nessun collegamento" non utilizzati sono tutti considerati ingressi del microcomputer poichè i loro stati logici non devono essere variati nel corso della prova. In questo modo i bit di ingresso e di uscita del microcomputer costituiscono le combinazioni che compongono la tabella della verità del chip genera-











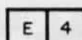


Figura 8-5. Byte di controllo della configurazione di I/O per il Quad AND Gate 74LS08.

ta dal software CHPTST. A titolo di esempio, al Figura 8-5 mostra come sono definiti i due byte di controllo della configurazione di I/O per il quad AND gate 74LS08. Nella Tabella 8-3 sono elencati per diversi sistemi logici combinatori appartenenti alla serie 74LS00 le coppie dei byte di controllo della configurazione di I/O.

Tabella 8-3. Byte di controllo della configurazione di I/O per sistemi logici combinatori diversi.

Circuito integrato	Parola di maschera	
	Byte Hi	Byte Lo
74LS02	C9	C9
74LS04	AB	EA
74LS05	AB	EA
74LS08	93	E4
74LS30	CF	C0
74LS32	93	E4
74LS42	87	FF
74LS125	93	E4
74LS139	8F	F8
74LS365	95	D4

SOFTWARE

In questa sezione ci proponiamo di esaminare brevemente il software associato al tester per circuiti integrati TTL, includendo altresì un listing ad esso relativo. Dovendo fare riferimento a questa struttura software, useremo d'ora in poi il nome CHPTST. Si noti che l'indirizzo assoluto corrispondente a ciascuna etichetta è reperibile nella Master Symbol Table (tabella dei simboli principali) riportata nell'Appendice A.

Codice oggetto	Codice sorgente		Commenti
3E03	CHPTST:	LD A,03H	; Inizializza il flip-flop di
			; abilitazione delle interruzioni del PIO
D30A		OUT (0AH),A	
D30B		OUT (0BH),A	
2A0300		LD HL,(MASKW)	; Predisponi la maschera per il dispositivo
010AFF		LD BC,0FF0AH	
ED41		OUT (C),B	; Predisponi la Porta A del PIO pel il modo 3
ED69		OUT (C),L	; Invia maschera per la Porta A
0C		INC C	; Indirizza la Porta B
ED41		OUT (C),B	; Predispone la Porta B del PIO per il modo 3
ED61		OUT (C),H	; Invia la maschera per la Porta B
	;		
31A00F	REF:	LD SP,CHPSTK	; Inizializza
			; lo Stack Pointer
DD210008		LD IX,REFIC	; Inizializza il puntatore alla
			; mappa del circuito integrato
			; di riferimento
010000		LD BC,0000H	; Inizializza la parola
			; del contatore
CD8806		CALL STORE	; Genera la tabella
			; di riferimento
00	ENDREF:	NOP	
	;		
31A00F	UNKN:	LD SP,CHPSTK	; Inizializza lo
			; Stack Pointer
DD21000C		LD IX,UNKIC	; Inizializza il puntatore della
			; mappa del circuito integrato
			; incognito

Codice oggetto	Codice sorgente		Commenti
010000		LD BC,0000H	; Inizializza la parola
			; del contatore
CD8806		CALL STORE	; Genera la tabella delle
			; uscite del circuito integrato
			; incognito
	;		
210008	COMPAR:	LD HL,REFIC	; Predisponi il confronto
			; mediante istruzione CPI
11000C		LD DE,UNKIC	; HL punta alla tabella di

```
                                            ; riferimento DE alla tabella
                                            ; del circuito integrato
                                            ; incognito
1A          NEXTB:   LD A,(DE)              ; Carica in accumulatore il byte
                                            ; sconosciuto in uscita
EDA1                 CP                     ; Confronta con (HL)
2037                 JR NZ,BAD              ; Se non vi è uguaglianza abbiamo
                                            ; a che fare con un C sbagliato
13                   INC DE                 ; Se uguale predisponi per la
                                            ; prova del prossimo byte
EA7D06               JP PE,NEXTB            ; Se il flag P/V = 1 procedi
                                            ; alla prova del prossimo byte
1833        GOOD:    JR START               ; Se il flag P/V = 0, BC è zero
                                            ; e tutti i byte sono
                                            ; stati provati
            ;
110000      STORE:   LD DE,0000H            ; Inizializza la parola di prova
2A0300      NTEST:   LD HL,(MASKW)          ; Carica HL con la parola di maschera
7B                   LD A,E                 ; Esegui in AND a 16 bit tra la
                                            ; parola di maschera e quella
                                            ; di prova
A5                   AND L
6F                   LD L,A
7A                   LD A,D
A4                   AND H
67                   LD H,A
7C          MASK:    LD A,H                 ; Verifica se il risultato dell'AND
                                            ; a 16 bit è uguale a 0

B5                   OR L
201B                 JR NZ,NXTWD            ; Se diverso da 0, passa alla parola
                                            ; da provare successiva

7B          TEST:    LD A,E                 ; Se uguale a 0 si tratta di una parola
                                            ; valida. Ponila in uscita verso il
                                            ; circuito integrato
D308                 OUT (08H),A
7A                   LD A,D
D309                 OUT (09H),A
```

Codice oggetto	Codice sorgente	Commenti
2A0300	LD HL,(MASKW)	; Prendi la parola di maschera ; relativa al circuito integrato
DB08	IN A,(08H),A	; Pon in ingresso il byte meno significativo ; del circuito integrato
A5	AND L	; Mascheralo
DD7700	LD (IX),A	; Memorizzalo
DD23	INC IX	; Aggiorna IX
DB09	IN A,(09H),	; Poni in ingresso il byte più significativo ; del circuito integrato
A4	AND H	; Mascheralo
DD7700	LD (IX),A	; Memorizzalo
DD23	INC IX	; Aggiorna IX
03	INC BC	; Aggiungi due al contatore
03	INC BC	

```
          ;
13        NXTWD:  INC DE        ; Vai a prelevare la prossima
                                ; parola di prova
7A                LD A,D
B3                OR E
20D3              JR NZ,NTEST   ; Se DE è diverso da zero, torna indietro
                                ; a prendere la prossima
                                ; parola di prova
C9                RET           ; Se DE è uguale a zero, è stata
                                ; generata una completa tabella
                                ; di valori in uscita
1800      BAD:    JR START      ; Circuito integrato sbagliato,
                                ; ricomincia

18AD      START:  JR UNKN       ; Salta alla routine di prova
                                ; dei circuiti integrati sconosciuti
```

Presentiamo qui sotto una mappa della memoria per le istruzioni, tabella dei dati e bufferi dei dati previsti da queste routine:

Etichetta in Memoria	*Descrizione*
da MASKW a MASKW + 1	Il byte L0 di controllo della configurazione, o di maschera, del circuito in prova, si trova alla locazione MASKW, seguito dal byte di maschera HI alla locazione MASKW+1. Nella Tabella 8-3 sono elencati questi byte di maschera relativi ad un buon numero di circuiti integrati.
CHPTST	E' il programma per il tester di circuiti integrati TTL.
da REFIC a REFIC+03FFH	E' l'area di memoria a Lettura/Scrittura contenente la tabella della verità generalizzata del circuito integrato di riferimento. (Un esempio del genere è illustrato nella Tabella 8-1). Alla tabella memorizzata in quest'area ci riferiremo nel seguito come alla tabella della risposta del RIFERIMENTO.
da UNKIC a UNCIK+03FFH	E' l'area di memoria a Lettura/Scrittura contenente la tabella della verità generalizzata del circuito integrato sconosciuto. (Un esempio del genere è illustrato nella tabella 8-1). Alla tabella memorizzata in questa area ci riferiamo nel seguito come alla tabella della risposta del componente SCONOSCIUTO.
CHPSTK	E' la locazione ove lo stack pointer inizialmente punta.

CHPTST comincia con l'inizializzazione delle due Porte C e D, del chip PIO N. 2. Si osservi come l'inizializzazione dei bit di I/O per entrambe le porte dipenda dalla parola di maschera caricata nella coppia di registri HL. In questa parola di maschera uno 0 logico sta ad indicare che il corrispondente bit della porta del PIO è un bit di uscita (uscita del PIO verso il circuito integrato da provare); un 1 logico indica invece che il corrispondente bit della porta del PIO è un bit di ingresso (ingresso al PIO dal

circuito integrato da provare). La Porta C corrisponde al byte di una maschera LO, mentre la Porta D corrisponde a quello HI.

Le routine REF e UNKN consistono in routine di inizializzazione dei registri, rispettivamente, per il circuiti integrato TTL di riferimento e per quello sconosciuto. Tali routine provvedono all'inizializzazione dello stack pointer, di un registro puntatore alla regione di memoria dove deve essere memorizzata la tabella della risposta del circuito integrato, e della coppia di registri BC, che sono utilizzati per contare il numero dei byte memorizzati nella tabella della risposta relativa al circuito integrato in prova. La routine COMPAR procede ad un confronto byte per byte delle tabelle della risposta relative al circuito integrato di riferimento e a quello sconosciuto. La parola di conteggio, memorizzata nella coppia di registri BC, è utilizzata per determinare il numero di byte da confrontare. Sul Nanocomputer vanno impostati due breakpoint alle locazioni GOOD e BAD (buono e cattivo) in modo da distinguere tra un circuito integrato buono ed uno cattivo (ossia difettoso). Al termine del confronto, sul display del Nanocomputer compare l'una o l'altra di queste due locazioni di breakpoint. Impostando in queste locazioni i breakpoint c'è tempo sufficiente tra un test e l'altro per la sostituzione dei circuiti integrati nella posizione di prova.

La parte più importante del software è costituita dalla subroutine STORE (carica in memoria), che, a sua volta include la routine MASK (maschera) adibita all'eliminazione delle parti non valide delle parole di prova.

Sono impiegati quattro registri a 16 bit:

Coppia di registri BC — E' la parola di conteggio, che conta il numero di byte memorizzati nella tabella della risposta relativa al circuito integrato in prova.

Coppia di registri DE — E' la parola di prova da scegliere, che varia da 0000 sino a FFFF. Un'operazione di AND con la parola di maschera permette al programma di determinare quale sia, tra le 65.536 possibili parole di prova, quella da inviare in uscita verso il circuito integrato in prova.

Coppia di registri HL — E' la parola di maschera relativa al circuito integrato in prova. In questa parola uno 0 logico indica un bit di uscita dal microcomputer, mentre un 1 logico indica un bit di ingresso nel microcomputer.

Registri indice IX — Punta alla tabella della risposta relativa al circuiti integrato in prova. Il prossimo byte di risposta è memorizzato nella locazione di memoria puntata dal registro IX.

La subroutine STORE provvede ad inizializzare a 0000 la parola di prova, caricare la parola di maschera, ed eseguire un'operazione di AND a 16 bit tra le parole di prova e di maschera. La routine MASK ha il compito di determinare se i bit del risultato dell'operazione di AND si trovano tutti e sedici allo stato logico 0 oppure no. In caso di risposta affermativa, la parola di prova, memorizzata nella coppia di registri DE, costituisce una parola di 16 bit valida per essere posta in uscita alle porte dati C e D del PIO N.2. Dopo l'operazione di uscita, due byte successivi sono introdotti nello accumulatore tramite il PIO, sottoposti all'operazione di maschera e quindi memorizzati nella tabella della risposta. La parola che costituisce il puntatore, memorizzata nel registro indice IX, è incrementata due volte nel corso di questo processo, come pure la parola del contatore. Per ultima, è incrementata la parola di prova e sottoposta a test con il valore 0000. Se la parola non è zero, la procedura di prova prosegue con un ritorno alla locazione NTEST.

Il ruolo svolto dalla parola di prova nel funzionamento della routine MASK può risultare meglio comprensibile ricorrendo ad un esempio. Si consideri un circuito integrato 74LS08 contraddistinto dalla parola di maschera 10010011 11100100

Tabella 8-4. Esempio di operazioni AND tra la parola di test e la parola di maschera.

Parola di prova		Parola di maschera del 74LS08		Prova · Maschera		Parola di prova valida?
00000000	00000000	10010011	11100100	00000000	00000000	si
00000000	00000001	10010011	11100100	00000000	00000000	si
00000000	00000010	10010011	11100100	00000000	00000000	si
00000000	00000011	10010011	11100100	00000000	00000000	si
00000000	00000100	10010011	11100100	00000000	00000100	no
00000000	00000101	10010011	11100100	00000000	00000100	no
00000000	00000110	10010011	11100100	00000000	00000100	no
00000000	00000111	10010011	11100100	00000000	00000100	no
00000000	00001000	10010011	11100100	00000000	00000000	si

ossia 93 E4 in codice esadecimale. Nella Tabella 8-4 sono state riportate alcune parole di prova, comprese tra 0000H e 0008H, con il risultato delle corrispondenti operazioni di AND tra queste parole e quella di maschera. Le prime quattro parole di prova sono parole valide per essere inviate al chip 74LS08; gli unici bit che cambiano sono infatti quelli corrispondenti alle posizioni PA0 e PA1 (si veda la Figura 8-5), che costituiscono i due ingressi della porta AND G1. Le quattro parole di prova successive sono eliminate in quanto il bit nella posizione PA2 presenta il valore logico 1. Poichè alla posizione PA2 non corrisponde un bit valido come uscita dal Microcomputer, in base alla parola di maschera riprodotta nella Figura 8-5 e nella Tabella 8-4, qualsiasi parola di prova contenente un 1 logico nella posizione PA2 può essere trascurata dal momento che un 1 o uno 0 in un bit di uscita è indifferente ai fini del risultato della prova. Lo stesso criterio vale per i bit delle posizioni PA5, PA6, PA7, PB0, PB1, PB4 e PB7: una parola di prova contenente un 1 logico in una qualsiasi di queste posizioni può, dunque, essere scartata. Ragionando in questo modo siamo in condizione di selezionare le sole 256 parole di prova valide tra le possibili 65.536 differenti parole di prova. Queste 256 parole di prova valide corrispondono alle 256 differenti combinazioni logiche degli otto ingressi del circuiti integrato 74LS08.

INTRODUZIONE AGLI ESPERIMENTI

Gli esperimenti che seguono sono studiati in modo da farvi raggiungere una certa esperienza nell'uso del circuito d'interfaccia e del programma del tester per circuiti integrati TTL. Nell'Esperimento N. 1 dovrete montare il circuito di interfaccia e provare due chip quad gate - il 74LS08 ed il 74LS32 - aventi byte di maschera identici. Nell'Esperimento N. 2 dovrete ottenere sperimentalmente il valore massimo della parola di conteggio, che conta il numero dei byte memorizzati nella tabella della risposta di riferimento relativa al circuito integrato in prova. Dovrete confrontare questo valore sperimentale con quello teorico per quel circuito integrato. Nello Esperimento N. 3 sarete invitati ad esaminare la tabella della risposta di riferimento relativa ad una porta di 74LS08.

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è quello di esaminare il funzionamento del tester per circuiti integrati TTL applicato ad un quad AND gate a 2 ingressi 74LS08 ed a un quad OR gate a 2 ingressi 74LS32.

Schema del circuito (Figura 8-6)

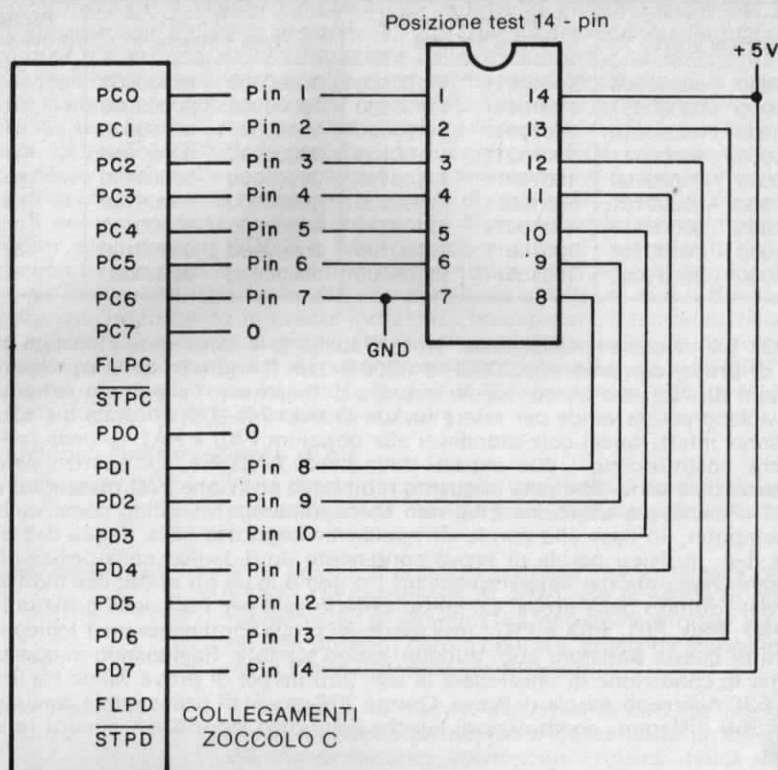


Figura 8-6. Schema elettrico del circuito per l'Esperimento N. 1.

Passo 1

Caricate il programma CHPTST.

Passo 2

Caricate le seguenti informazioni nella memoria a lettura/scrittura a partire da MASKW. (Consultare l'Appendice A, Tabella A-1, per l'indirizzamento assoluto).

Indirizzo in memoria	Contenuto	Descrizione
MASKW	E4	Byte di maschera L0 per 74LS08
MASKW + 1	93	Byte di maschera Hi per 74LS08

Passo 3

Premete il tasto BRK per entrare nel modo di breakpoint, quindi introducete i seguenti indirizzi di breakpoint valendovi del tasto INC:

Locazione di memoria	Commenti	
ENDREF	ENDREF:	Breakpoint al completamento di REF
GOOD	GOOD:	Breakpoint per un integrato "buono"
BAD	BAD:	Breakpoint per un integrato "cattivo"

Sarà bene ricordare che l'ordine di introduzione di breakpoint è indifferente. Introdotti questi breakpoint, premete il tasto BRK per uscire dal modo di breakpoint.

Passo 4

Disporre il circuito integrato di riferimento, 74LS08 nella posizione di prova ed iniziare l'esecuzione del programma CHPTST. Dopo un breve intervallo di tempo vedrete comparire l'indirizzo di breakpoint, ENDREF sul display della tastiera.

A questo punto avete generato la tabella della risposta del RIFERIMENTO relativa all'integrato di riferimento, un 74LS08. Staccate le linee dell'alimentazione e di massa dal 74LS08 e toglietelo quindi dalla posizione di prova.

Passo 5

Disponete il circuito integrato "sconosciuto" (il 74LS32) nella posizione di prova e collegategli le linee di alimentazione e di massa. Si noti che il 74LS32 è caratterizzato dalla medesima parola di maschera del 74LS08. Cominciate l'esecuzione della routine UNKN premendo il tasto GO.

Noi abbiamo osservato che l'esecuzione del programma si interrompe al breakpoint BAD visualizzando sulla tastiera del Nanocomputer tale indirizzo. Questo è il breakpoint che sta ad indicare che la tabella della risposta del componente SCONOSCIUTO non si accorda con quella del RIFERIMENTO. Nel caso nostro non vi è motivo di stupirci in quanto abbiamo utilizzato due integrati differenti. Se, invece, i due integrati fossero stati del medesimo tipo (ad esempio due 74LS08) e fossimo sicuri che l'integrato di RIFERIMENTO era sicuramente buono, saremmo in grado, a tal punto, di concludere che il 74LS08 "SCONOSCIUTO" era un integrato difettoso.

Passo 6

Staccate le linee di alimentazione e massa del 74LS32 e toglietelo dalla posizione di prova. Inserite al suo posto il 74LS08 e collegategli le linee di alimentazione e massa. Riprendete l'esecuzione premendo il tasto GO.

Noi abbiamo osservato che l'esecuzione del programma si interrompe al breakpoint GOOD visualizzando tale indirizzo sulla tastiera del Nanocomputer. Abbiamo cioè generato una tabella di risposta del componente "SCONOSCIUTO" identica alla tabella della risposta del RIFERIMENTO. il che non dovrebbe sorprenderci dal momento che i chip utilizzati per generare le due tabelle sono del tutto identici.

ESPERIMENTO N. 2

Scopo

Lo scopo di questo esperimento è quello di determinare il valore massimo della parola di conteggio, che fornisce il numero di byte memorizzati nella tabella di risposta del circuito integrato in prova.

Passo 1

Per questo esperimento non occorre nessun circuito integrato. E' sufficiente caricare in memoria parole di maschera differenti, passando quindi a determinare il valore della coppia di registri BC allorchè l'esecuzione del programma si arresta al breakpoint ENDREF. L'esecuzione deve sempre cominciare da CHPTST.

Passo 2

Caricate la parola di maschera 93 E4, relativa ad un chip 74LS08 nelle locazioni di memoria MASKW e MASKW+1, il byte LO (E4) per primo e poi quello HI (93). Eseguite il programma del tester a partire da CHPTST e attendete la comparsa sul display della tastiera dell'indirizzo di breakpoint, ENDREF.

Passo 3

Posizionate l'indicatore luminóso di selezione in BC. Quale valore osservate?

Noi abbiamo osservato 0200, corrispondente ad un conteggio di 512 byte nella tabella di risposta del RIFERIMENTO relativa al 74LS08. La ragione di questo sarà spiegata nel Passo 4.

Passo 4

Quanti byte prevedete che saranno memorizzati nella tabella di risposta del RIFERIMENTO relativa al chip 74LS08, caratterizzata dalla parola di maschera 93 E4? Giustificate la vostra previsione.

Per il 74LS08 noi abbiamo calcolato che occorre uno spazio di memoria pari a 512 byte. La parola di maschera, scritta in forma binaria è

1001 0011 1110 0100

In questa parola sono prsenti 8 bit di valore logico 0, il che significa che nella tabella della verità del 74LS08 vi sono 256 combinazioni. Per ogni combinazione della tabella della verità devono essere memorizzati due byte, un byte LO (Porta C) ed uno HI (Porta D). Sono dunque necessari in totale 512 byte, corrispondenti in esadecimale a 0200, il che coincide con quanto compare sul display del Nanocomputer visualizzando il contenuto della coppia di registri BC al breakpoint ENDREF.

Passo 5

Caricate la parola di maschera C9 C9, corrispondente al chip 74LS02, nelle rispettive locazioni di memoria MASKW e MASKW+1. Cominciate l'esecuzione a CHPTST, attendete sino a che non appaia l'indirizzo di breakpoint ENDREF e, quindi, passate ad esaminare il contenuto della coppia di registri BC in occasione di questo breakpoint. Che valore osservate? Il valore che avete osservato corrisponde a quello che attendevate?

Noi abbiamo osservato un valore di 0200, corrispondente alla memorizzazione di 512 byte nonchè al valore da noi calcolato.

Passo 6

Ripetete il Passo 5, questa volta, però, con la parola di maschera CF C0, corrispondente al chip 74LS30. Quale parola di conteggio osservate all'apparire dell'indirizzo di breakpoint ENFDREF?

Noi abbiamo osservato una parola di conteggio di 0200. Il chip 74LS30 consiste, infatti, in una porta NAND a 8 ingressi, per cui, oltre ai due pin per l'alimentazione, vi sono 8 ingressi, un'uscita e 3 pin "nessun collegamento". Volendo provare correttamente questo chip, i pin "da non collegare" devono essere posti a massa oppure a +5V. Otto ingressi corrispondono a 256 combinazioni delle tabelle della verità del sistema 74LS30, per cui, siccome ogni combinazione comporta la memorizzazione di due byte, nella tabella di risposta sono stati memorizzati in totale 512 byte (0200 in esadecimale).

Passo 7

Quanti byte sarebbero memorizzati per un Dual Decodificatore/Demultiplexer 1 a 4 74LS139?

Con 6 ingressi differenti sarebbero memorizzati 128 byte, corrispondenti in esadecimale alla parola di conteggio 0080, quella, appunto, osservata durante l'esperimento.

Passo 8

Esaminate attentamente le caratteristiche fornite per il 74LS42 nella Tabella 8-2. Quante locazioni di memoria sono necessarie per la memorizzazione della Tabella della risposta relativa al circuito integrato di RIFERIMENTO 74LS42? Quante locazioni sono necessarie alla memorizzazione della tabella di risposta del componente SCONOSCIUTO relativa al circuito integrato 74LS42?

Con 4 ingressi per l'integrato di RIFERIMENTO è necessario il doppio di 2^4 locazioni (ossia 32), ed esattamente lo stesso numero di byte è necessario a memorizzare la tabella di risposta del componente SCONOSCIUTO. Ne deriva che il registro BC dovrebbe contenere 0020 dopo che sia stata generata la tabella di risposta del RIFERIMENTO.

Passo 9

Esaminate attentamente le caratteristiche della porta AND-OR-INVERT ad Ingresso 3-2-2-3 74LS54 nella Tabella 8-2. Si noti come i pin di ingresso di questo chip siano 10. Quante sono le locazioni di memoria necessarie a memorizzare la tabella di risposta del circuito integrato 74LS54 di RIFERIMENTO? Quanti sono i byte necessari a memorizzare la tabella di risposta del componente SCONOSCIUTO relativa al 74LS54?

Con dieci ingressi nella tabella della verità del sistema, per la tabella della risposta di RIFERIMENTO è necessario il doppio di 2^{10} byte (ossia 2048). Lo stesso numero di byte, 2048, è necessario a memorizzare la tabella di risposta del componente SCONOSCIUTO relativa ad un 74LS54. Solo per memorizzare le due tabelle di risposta abbiamo dunque bisogno in totale di 4096 byte. Poichè la mappa della memoria che si riferisce alla routine CHPTST mostra che per ogni tabella di risposta disponiamo di un massimo di 1024 byte, volendo provare un integrato, 74LS54, dovremmo apportare alcune modifiche.

ESPERIMENTO N. 3

Scopo

Lo scopo di questo esperimento è quello di esaminare la tabella di risposta del componente di RIFERIMENTO relativa ad un Quad AND gate a due ingressi 74LS08.

Passo 1

Proponiamo di esaminare una singola porta del 74LS08. Dalla Figura 8-1, schema della configurazione dei pin, si può osservare che l'uscita del microcomputer andrà ai Pin 1 e 2 e che la risposta inviata al microcomputer dal 74LS08 comparirà in corrispondenza del Pin 3. Per questo esperimento considereremo, perciò, soltanto i Pin 1 e 2 come pin di uscita del microcomputer, cosicchè la parola di maschera sarà:

1111 1111 1111 1100 ossia FF FC in codice esadecimale

Caricate tale parola di maschera (byte HI = FF) (byte LO = FC) rispettivamente alle locazioni MASKW e MASKW+1. Verificate che siano predisposti gli opportuni breakpoint come nel Passo 3 dell'Esperimento Numero 1.

Passo 2

Inserite un chip 74LS08 nella posizione di prova, collegando i pin di alimentazione e di massa rispettivamente a +5 e 0. Poichè il 74LS08 è un integrato a 14 pin, dovete accertarvi che sia PA7 che PB0 siano posti a massa. Date corso all'esecuzione del programma partendo da CHPTST. Dopo la comparsa del breakpoint ENDREF sul display del Nanocomputer esaminate il contenuto del registro BC.
Noi abbiamo osservato 0008 come contenuto del registro BC.
Tale contenuto è, d'altronde, esattamente quello che ci attendevamo poichè i pin che sono stati definiti come uscita dal microcomputer verso il 74LS08 sono solo 2 (1 e 2). Le combinazioni possibili nella tabella della verità relativa a questa singola porta sono dunque quattro e, poichè per ognuna di esse sono necessari due byte, i byte occorrenti per la tabella di risposta del RIFERIMENTO sono 8.

Passo 3

Procederemo adesso all'esame della tabella di risposta del RIFERIMENTO, la cui locazione iniziale nella memoria a lettura/scrittura è all'indirizzo 0800H. Esaminate e registrate il contenuto di 8 successive locazioni di memoria a partire da 0800H.

Noi abbiamo osservato questi risultati:

Tabella 8-5. Tabella di risposta del componente di RIFERIMENTO relativa ad una singola porta di 74LS08.

Locazione di memoria	Contenuto	Equivalente binario
REFIC	38	0011 1000
REFIC + 1	FE	
REFIC + 2	38	0011 1000
REFIC + 3	FE	
REFIC + 4	38	0011 1000
REFIC + 5	FE	
REFIC + 6	3C	0011 1100
REFIC + 7	FE	

Poichè i byte contenuti alle locazioni, REFIC+1, REFIC+3, REFIC+5, e REFIC+7, non hanno nulla a che fare con la porta che stiamo esaminando potremmo fare a meno di considerarli.

Passo 4

Da un esame delle istruzioni del programma CHPTST si vede che l'ordine nel quale sono generati i valori posti in uscita verso i Pin 1 e 2 segue l'andamento qui riportato:

Parola di prova	Pin 1	Pin 2	Pin 3
0000	0	0	0
0001	1	0	0
0002	0	1	0
0003	1	1	1

Nella colonna indicata con PIN 3 abbiamo illustrato lo stato logico dell'ingresso al microcomputer proveniente dal 74LS08 per ciascuna delle parole di prova.

Questo ingresso al microcomputer proveniente dal PIN 3 del 74LS08 compare come terzo bit di ciascuno dei byte alle locazioni REFIC, REFIC+2, REFIC+4 e REFIC+6 della tabella di risposta del RIFERIMENTO. Tali valori, appunto, sono stati evidenziati nella Tabella 8-5. Poichè il 74LS08 è una porta AND, la procedura da noi eseguita ci ha permesso di verificare che i valori della nostra tabella di risposta del RIFERIMENTO sono corretti.

CAPITOLO 9

IL CIRCUITO CONTATEMPI - CONTAEVENTI CTC Z80

INTRODUZIONE

Il dispositivo Contatempi-Contaeventi (Counter-Timer Circuit) CTC Z80, al pari del dispositivo PIO Z80, appartiene ad una famiglia di circuiti integrati realizzati appositamente per agevolare l'interfacciamento della CPU Z80. Il CTC provvede alle funzioni di temporizzazione e di conteggio degli eventi mediante quattro canali indipendenti ad otto bit ed è direttamente collegabile al bus dei dati dello Z80. Il CTC è un componente programmabile: ciascuno dei suoi canali può essere configurato, in modo indipendente, per operare in Counter Mode (modo contaeventi, contatore) oppure in Timer Mode (modo contatempi, temporizzatore). Il Counter Mode fa sì che il CTC accetti, contandoli, degli impulsi (eventi) provenienti da una sorgente esterna. Il CTC può anche essere predisposto in modo da inviare una interruzione (in Modo 2) alla CPU quando è stato ricevuto un certo numero di impulsi. Tanto il vettore di interruzione, che il numero di impulsi, ricevuti i quali viene generata l'interruzione, sono definibili da software. Nel Timer Mode, la funzione del CTC è essenzialmente quella di contare gli impulsi del clock di sistema, Φ . Anche in questo secondo caso la CPU può programmare il CTC in modo che questo generi un segnale di interruzione dopo un determinato numero di impulsi. Poichè il periodo del clock di sistema è noto e di elevata stabilità (si tratta di un oscillatore al quarzo!), il CTC può essere predisposto in modo da inviare segnali di interruzione alla CPU ad intervalli di tempo rigorosamente definiti, misurando così gli intervalli di tempo come sarebbero misurati da un normale orologio. A seconda di come il CTC è programmato, questo orologio può avere una risoluzione dell'ordine di millisecondi, oppure di secondi.

Passiamo ora brevemente in rassegna le principali caratteristiche del CTC:

- Il CTC è fabbricato in tecnologia MOS "silicon-gate depletion load", canale N, e racchiuso in un contenitore DIP a 28 piedini. Sono necessari unicamente un'alimentazione di +5 volt ed un segnale clock di 5 volt monofase.
- Gli ingressi e le uscite sono tutti compatibili TTL.
- Ciascuno dei quattro canali può essere definito ad operare in Counter oppure in Timer Mode.
- Per ciascun canale, sia nell'uno che nell'altro modo di funzionamento, la CPU può leggere il contenuto di un Registro Down-Counter (Contatore in decremento), può leggere cioè il numero di impulsi che devono ancora essere ricevuti perchè il contatore vada a zero.
- Un registro Costante di Tempo può ricaricare automaticamente il Down Counter, azzerato, in entrambi i modi di funzionamento.
- E' possibile, da programma, definire se gli impulsi di clock o i segnali di conteggio degli eventi sono attivi (decrementano cioè il contatore) in corrispondenza del loro fronte positivo oppure di quello negativo.

- La generazione di una interruzione quando il conteggio giunge a zero avviene sotto il controllo del software.
- Contiene dei circuiti che permettono senza bisogno di logica esterna la gestione vettorizzata delle interruzioni all'interno di una catena di priorità.
- Tre dei canali sono dotati di uscite Conteggio Zero/Termine Tempo Impostato in grado di pilotare dei transistori Darlington.

OBIETTIVI

Al termine di questo capitolo sarete in grado di:

- Avere ben chiara la funzione di tutte le caratteristiche del CTC che abbiamo appena elencato.
- Verificare e studiare sperimentalmente nel corso di esercitazioni pratiche tali caratteristiche.
- Sapere interpretare il *Manuale Tecnico del CTC Z80,* pubblicato dalla Zilog e dalla SGS-ATES.

ASPETTI GENERALI DEL DISPOSITIVO CTC Z80

Questa sezione è dedicata alla descrizione della configurazione dei piedini del CTC, ad un sommario esame dei suoi modi di funzionamento e ad alcuni cenni sul modo di programmarlo. Come nel caso del PIO, abbiamo di proposito voluto che questa sezione risultasse concisa, in modo che vi possiate accingere quasi subito ad effettuare gli esperimenti, nel corso dei quali vi saranno via via presentate nei particolari le modalità di impiego del CTC.

La Figura 9-1 riporta la configurazione dei piedini del CTC, delle cui funzioni diamo ora la descrizione.

La descrizione dei piedini, nella forma qui adottata, è tratta dal *Manuale Tecnico* del *CTC Z80* pubblicato a cura della SGS-ATES:

D7-D0 — Bus dei Dati della CPU Z80 (bidirezionale, tre stati).

Questo bus serve al trasferimento di tutte le informazioni (dati e comandi) tra la CPU Z80 e il CTC Z80. Il bus è ad otto bit, il meno significativo dei quali è D0.

CS1-CS0 — Channel Select: Selezione del Canale (ingressi, attivi alti).

Attraverso questi piedini, su cui viene presentato un codice binario d'indirizzo a 2 bit, viene selezionato uno dei quattro canali del CTC per le operazioni di Lettura (Input) o di Scrittura (Output). (Si veda la tabella della verità riportata di seguito).

	CS1	CS0
Can. 0	0	0
Can. 1	0	1
Can. 2	1	0
Can. 3	1	1

$\overline{CE}$ — Chip Enable: Abilitazione del dispositivo (ingresso, attivo basso)

Un livello basso su questo piedino abilita il CTC (Ciclo di Scrittura in I/O) ad accettare, dal Bus dei dati della CPU Z80, parole di controllo, Vettori di Interruzione o parole di definizione della costante di tempo oppure (Ciclo di Lettura in I/O) a trasmettere alla CPU il contenuto del Down Counter. Nella maggior parte delle applicazioni, questo segnale è generato, tramite decodifica, dagli 8 bit meno significativi

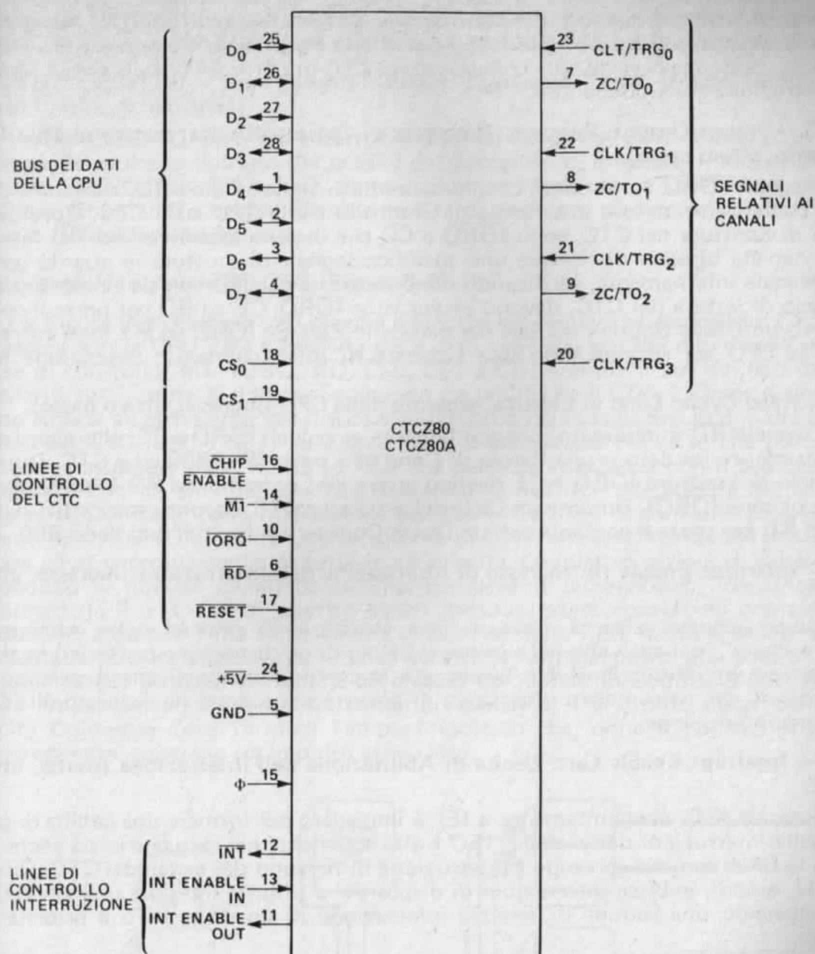


Figura 9-1. Configurazione dei piedini del CTC.

del bus degli indirizzi per ciascuno dei quattro indirizzi di I/O che corrispondono ai quattro canali del CTC.

Clock (Φ) – Clock del Sistema (ingresso)

Questo clock a fase singola è utilizzato dal CTC per la sincronizzazione interna di alcuni segnali.

$\overline{\text{M1}}$ – Machine Cycle One: Ciclo Macchina Uno, generato dalla CPU (ingresso, attivo basso).

Se $\overline{\text{M1}}$ è attivo ed $\overline{\text{RD}}$ anche, la CPU sta procedendo al prelievo (fetch) di un'istruzione in memoria. Se $\overline{\text{M1}}$ è attivo e $\overline{\text{IORQ}}$ anche, vuol dire invece che la CPU sta pro-

cedendo al riconoscimento di un'interruzione. Se uno dei canali del CTC ha inoltrato una richiesta di interruzione e se tale interruzione è quella a più alta priorità all'interno della daisy chain viene così comunicato al CTC di porre sul bus dei dati il Vettore d'Interruzione corrispondente.

$\overline{\text{IORQ}}$ — Input/Output Request: Richiesta di Ingresso/Uscita, generato dalla CPU (ingresso, attivo basso).

Il segnale $\overline{\text{IORQ}}$ è utilizzato, congiuntamente ai segnali $\overline{\text{CE}}$ ed $\overline{\text{RD}}$, allorchè si desidera trasferire un dato o una Parola di Controllo tra la CPU ed il CTC. Durante un Ciclo di Scrittura nel CTC, sono $\overline{\text{IORQ}}$ e $\overline{\text{CE}}$ che devono essere veri ed $\overline{\text{RD}}$ falso. Il CTC non ha bisogno di ricevere uno specifico segnale di scrittura in quanto genera tale segnale internamente, verificando semplicemente che $\overline{\text{RD}}$ non sia valido. Durante un ciclo di lettura del CTC, devono essere attivi $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ ed $\overline{\text{RD}}$ per porre il contenuto di un Down Counter sul bus dei dati dello Z80. Se $\overline{\text{IORQ}}$ ed $\overline{\text{M1}}$ sono entrambi veri, la CPU sta riconoscendo una richiesta di interruzione (v. descrizione M1).

$\overline{\text{RD}}$ — Read Cycle: Ciclo di Lettura, generato dalla CPU (ingresso, attivo basso).

Il segnale $\overline{\text{RD}}$ è impiegato, congiuntamente ai segnali $\overline{\text{IORQ}}$ e $\overline{\text{CE}}$, allorchè si desidera trasferire un dato o una Parola di Controllo tra la CPU Z80 ed il CTC. Durante un Ciclo di Scrittura $\overline{\text{IORQ}}$ e $\overline{\text{CE}}$ devono essere veri entrambi ed $\overline{\text{RD}}$ falso (v. descrizione piedino $\overline{\text{IORQ}}$). Durante un Ciclo di Lettura del CTC, devono essere attivi $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ ed $\overline{\text{RD}}$ per porre il contenuto di un Down Counter sul bus dei dati dello Z80.

IEI — Interrupt Enable In: Ingresso di Abilitazione dell'Interruzione (ingresso, attivo alto).

Questo segnale serve a formare una struttura di priorità delle interruzioni "daisy-chain", nel caso che nel sistema vi sia più di un dispositivo periferico in grado di generare un'interruzione. Un livello alto in corrispondenza di questo piedino sta ad indicare che nessun altro dispositivo di priorità più elevata ha richiesto alla CPU Z80 una interruzione.

IEO — Interrupt Enable Out: Uscita di Abilitazione dell'Interruzione (uscita, attivo alto).

Il segnale IEO, congiuntamente a IEI, è impiegato per formare una catena di priorità delle interruzioni daisy-chain. IEO è alto solamente nel caso che lo sia anche IEI e che la CPU non stia servendo l'interruzione di nessuno dei canali del CTC. Questo segnale, quindi, inibisce interruzioni di dispositivi a priorità inferiore quando la CPU sta eseguendo una routine di servizio interruzione di un dispositivo a priorità più elevata.

$\overline{\text{INT}}$ — Interrupt Request: Richiesta di interruzione (uscita, open drain, attivo basso).

Questo segnale diventa attivo quando uno dei canali del CTC, abilitato da programma a richiedere interruzioni, abbia il proprio Down Counter in condizione di conteggio zero.

$\overline{\text{RESET}}$ — Reset (ingresso, attivo basso).

Per tutti i canali questo segnale arresta l'operazione di conteggio e pone a zero i bit di abilitazione interruzione nei registri di controllo, disabilitando, perciò, le interruzioni generate dal CTC. Le uscite ZC/TO e $\overline{\text{INT}}$ passano nello stato non-attivo, IEO seguirà lo stato di IEI ed i driver di uscita del bus dei dati del CTC passano nello stato ad alta impedenza.

CLK/TRG3 ÷ CLK/TRG0 — External Clock/Timer Trigger: Clock Esterno/Segnale di partenza del contaeventi (ingresso, attivo basso o alto, a scelta dell'utente).

I piedini CLK/TRG sono quattro, uno per ogni canale del CTC. Nel Counter Mode, ogni fronte attivo presente su questi piedini decrementa il Down Counter. Nel Timer

Mode, un fronte attivo sul piedino CLK/TRG dà inizio alla funzione di conteggio tempi. Il fronte attivo può essere quello di salita oppure di discesa a scelta dell'utente.

ZC/TO2 ÷ ZC/TO0 – Zero Count/Timeout: Conteggio Zero/Termine Tempo impostato (uscita, attivo alto).

I piedini ZC/TO sono tre e corrispondono ai canali del CTC da 0 a 2. In conseguenza del numero limitato dei piedini del contenitore, il canale 3 non ha invece il piedino ZC/TO. Per ogni canale, tanto in Counter Mode che in Timer Mode, allorchè il Down Counter si è decrementato sino a zero, su questo piedino compare un impulso con fronte di salita attivo.

Descrizione funzionale del CTC (Figura 9-2)

La Figura 9-2 presenta uno schema a blocchi funzionale del dispositivo CTC. L'interfaccia con la CPU Z80 è formata dalle otto linee del bus dei dati e dalle seguenti linee di controllo: $\overline{M1}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{CS0}$, $\overline{CS1}$ e $\overline{CE}$. Tramite il bus dei dati vengono trasferiti tutti i byte di dato e di comando tra la CPU ed il CTC. Le linee di controllo sono adibite all'attivazione del dispositivo CTC ($\overline{CE}$), alla selezione di uno dei quattro canali del CTC (CS0 e CS1) ed a specificare e sincronizzare il verso che il flusso dei dati deve assumere ($\overline{M1}$, $\overline{IORQ}$ ed $\overline{RD}$). La parte di interfaccia con il bus della CPU comunica con gli altri blocchi interni tramite il bus dei dati interno del CTC. La Logica di Controllo delle Interruzioni gestisce le tre linee di controllo delle interruzioni: IEI, IEO e $\overline{INT}$. Una Logica Interna di Controllo sovrintende alla sincronizzazione ed al coordinamento di tutte le varie parti. Le quattro sezioni di canale corrispondono ai quattro canali contatempi/contaeventi indipendenti, identificati con i numeri da 0 a 3. Questi quattro canali possono essere considerati come quattro dispositivi adiacenti nella solita catena di priorità dello Z80, in cui al canale numero 0 spetta la priorità più elevata. I canali accettano, come ingresso, una linea di Clock/Trigger su cui vengono presentati gli impulsi da contare in Counter Mode. A causa delle limitazioni del contenitore, soltanto i canali 0, 1 e 2 sono dotati di linee di uscita Conteggio Zero/Termine Tempo Impostato che, dopo il numero di impulsi programmato, generano un impulso attivo alto.

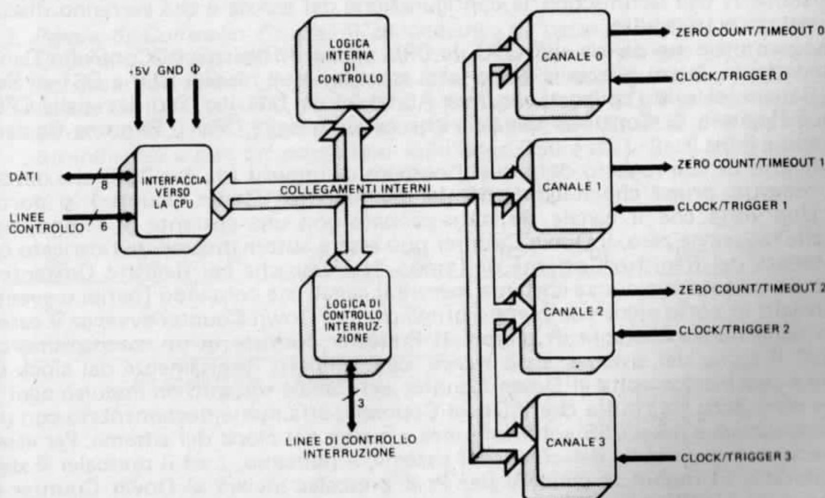


Figura 9-2. Schema a blocchi funzionale del CTC.

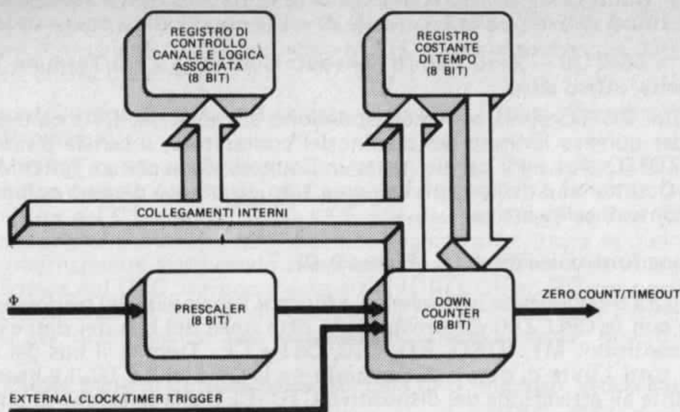


Figura 9-3. Schema a blocchi funzionale di un canale del CTC.

La Figura 9-3 presenta uno schema a blocchi funzionale relativo ad uno dei quattro canali del CTC. Gli elementi funzionali principali sono rappresentati da due registri, due contatori, la logica di controllo e un bus interno. Il Registro di Controllo Canale (e logica associata) contiene un registro a 8 bit che viene caricato dalla CPU con un valore che definisce il modo di operare del canale.

Il contenuto di questo registro determina se il canale si trova in Counter Mode o in Timer Mode, se le interruzioni del canale sono disabilitate o abilitate, se il clock del canale deve essere ridotto (solo in Timer Mode), tramite un prescaler (riduttore), per 256 o per 16, se è il fronte di salita o quello di discesa che fa da trigger del contatempi (in Timer Mode) o che fa da clock per il contaeventi (in Counter Mode), ed altri parametri che definiscono la configurazione del canale e che verranno discussi più in dettaglio in seguito.

Per programmare un canale del CTC, la CPU carica il Registro di Controllo Canale corrispondente, la cui selezione viene fatta tramite i due piedini CS0 e CS1 di Selezione Canale (di solito collegati alle linee A0 ed A1 del bus degli indirizzi della CPU). Dopo il Registro di Controllo canale viene caricato dalla CPU il Registro Costante di Tempo a 8 bit.

Il contenuto di tale registro definisce il numero di impulsi (da 1 a 256) che devono essere ricevuti prima che il contatore in decremento (Down Counter) si porti a zero. Una volta che il canale sia stato caricato con una costante di tempo, ogni volta che raggiunge zero, il Down Counter può essere automaticamente ricaricato con il contenuto del Registro Costante di Tempo. Nel caso che nel Registro Costante di Tempo sia caricata una nuova costante mentre il canale sta contando (tempi o eventi), il conteggio in corso viene completato prima che nel Down Counter avvenga il caricamento della nuova costante di tempo. Il Prescaler consiste in un meccanismo che "divide" il clock del sistema. Esso riceve, cioè, impulsi direttamente dal clock del sistema e passa a sua volta al Down Counter del Canale soltanto un impulso ogni 16 oppure ogni 256. Ne risulta che il Down Counter potrà essere decrementato con una frequenza che è o 16 o 256 volte inferiore a quella del clock del sistema. Per essere più precisi, se il periodo del clock del sistema, è poniamo, t ed il prescaler è stato programmato in modo da dividere per P, il prescaler invierà al Down Counter un impulso ogni t x P unità di tempo. Il rapporto di divisione (16 o 256) è definito dall'utente, da programma. Il Prescaler può essere "collegato" solo al clock del sistema, ed è perciò impiegato esclusivamente in Timer Mode, mentre in Counter

Mode, gli impulsi esterni sono portati direttamente all'ingresso del Down Counter. Tanto in Counter quanto in Timer Mode, allorchè il Down Counter giunge a zero, sul piedino Zero Count/Timeout viene generato un impulso di segnalazione. Poichè le limitazioni dovute al package non consentono che ciascun canale sia dotato della propria linea di Count Zero/Timeout, il canale 3 è privo del piedino corrispondente. Tutti i canali del CTC, comunque, possono essere programmati in modo da inviare un segnale di interruzione alla CPU (linea $\overline{INT}$) quando si sia verificata la condizione di Conteggio Zero o di Termine tempo impostato.

Come si programma il CTC

Programmare il dispositivo CTC vuol dire specificare tre importanti tipi di parametri:

1. *Vettore di Interruzione:* il CTC è previsto per generare interruzioni con la CPU Z80 nel Modo di Interruzione 2. Quando perciò alla CPU perviene un segnale di interruzione dal CTC, quest'ultimo deve presentare un codice di identificazione dispositivo (device identification = Id). (Per un'analisi della procedura di interruzione in Modo 2 della CPU Z80 si veda il Capitolo 6). Tale codice di identificazione è interpretato come il byte meno significativo di un indirizzo verso una tabella dei vettori, che, a sua volta, punta all'inizio di una routine di servizio dell'interruzione. Il codice di identificazione o meglio i suoi cinque bit di ordine superiore, sono caricati nel CTC scrivendo un byte all'indirizzo della porta di I/O corrispondente al canale 0 del CTC. Per segnalare al CTC che si tratta di un byte che definisce il vettore di interruzione il bit D0 di tale byte deve essere posto a 0. I bit D7, D6, D5, D4, D3 e D0 sono quindi caricati tutti nel Registro Vettore di Interruzione, mentre il contenuto dei bit D2 e D1 è controllato automaticamente dalla logica di Controllo delle Interruzioni del CTC. Quando il canale che ha richiesto l'interruzione deve porre il vettore di interruzione sul bus dei dati della CPU Z80, la logica di Controllo delle Interruzioni del CTC inserisce automaticamente un codice binario nei bit D1 e D2, identificando così quale tra i quattro canali del CTC deve essere servito. La Figura 9-4 illustra la corrispondenza tra il contenuto di D1 e D2, i canali del CTC ed il vettore di interruzione che ne risulta.

2. *Parola di Controllo Canale:* Il caricamento da parte della CPU della Parola di Controllo Canale nel Registro di Controllo Canale avviene effettuando una normale operazione di scrittura in una porta al corrispondente indirizzo del canale del CTC; il CTC, se esistono più dispositivi di I/O, è selezionato portando attivo $\overline{CE}$. Nel CTC così abilitato la selezione del canale si effettua mediante un indirizzo a due bit presentato sulle linee CS0 e CS1. Se il byte sul bus dei

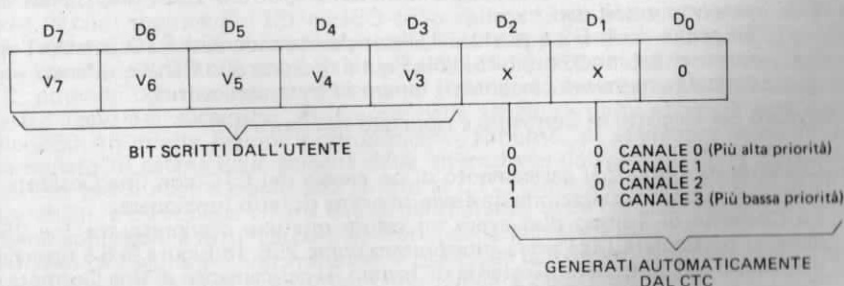


Figura 9-4. Codifica dei bit D1 e D2 nel vettore di interruzione del CTC.

dati della CPU conterrà il bit D0 posto al valore 1, il canale del CTC riconoscerà nel byte una Parola di Controllo Canale e la caricherà nel proprio Registro di Controllo Canale. I bit da D1 a D7 hanno il seguente significato:

D7: **Interrupt Enable:** *Abilitazione dell'Interruzione* – Se D7 è posto a 1, il canale del CTC è abilitato a generare un'interruzione tutte le volte che il Down Counter arriva a zero nel conteggio. Se D7 è posto a 0, non si genererà invece nessuna interruzione. Ove si vogliano abilitare le interruzioni, è necessario, prima di avviare qualsiasi operazione, caricare nel Registro Vettore di interruzione del dispositivo CTC il vettore di interruzione voluto.

D6: **Mode Select:** *Selezione del Modo* – Se D6 è posto a 1, è il Counter Mode ad essere selezionato, mentre in caso contrario, risulta selezionato il Timer Mode.

D5: **Prescaler Range:** *Valore del Prescaler* – Del bit D5 ci si vale soltanto in Timer Mode, al fine di predisporre il Prescaler a dividere per 16 oppure per 256 il clock di sistema. Il valore 1 corrisponde a 256 mentre quello 0 corrisponde a 16.

D4: **Slope:** *Fronte* – Questo bit specifica quale fronte del clock farà partire il conteggio del temporizzatore (in Timer Mode) o farà decrementare il Down Counter (in Counter Mode). Il valore 1 sceglierà il fronte positivo, mentre 0 il negativo.

D3: **Trigger** – Di questo bit ci si vale soltanto in Timer Mode, per specificare quale segnale di partenza (trigger), farà partire il contatempi. Il valore 1 stabilisce che il trigger sarà esterno (linea External Clock/Timer Trigger) Un valore 0 fa sì che il contatempi cominci invece a funzionare in corrispondenza del fronte di salita della fase T2 del ciclo di macchina immediatamente successivo a quello in cui avviene il caricamento della costante di tempo. Nel caso di selezione del trigger esterno, il Prescaler è decrementato dopo che sono trascorsi due cicli di clock, se è soddisfatta la condizione per il tempo di set-up (assestamento, 130 nanosecondi), del segnale di Clock/Trigger (CLK/TRG). In pratica, CLK/TRG deve diventare alto almeno 130 nanosecondi prima del primo fronte attivo di Φ, così che il ritardo tra il trigger esterno e l'attivazione del Prescaler sia di due cicli di clock. In caso contrario, il ritardo sarà di tre cicli di clock.

D2: **Load Time Constant:** *Caricamento della Costante di Tempo* – Se il bit D2 è posto a 1, il canale del CTC è pronto a ricevere, come parola successiva ad esso indirizzata, il byte che definisce la costante di tempo.

D1: **Reset Channel:** *Reset del Canale* – Quando il bit D1 è posto a 1, il canale interrompe il conteggio. I bit del Registro di Controllo Canale non vengono modificati.
Se anche il bit D2 è posto a 1, il canale riprende automaticamente l'operazione, nel modo definito dalla Parola di Controllo Canale, quando verrà caricata una nuova costante di tempo (il byte successivo).

Il formato del Registro di Controllo è riportato in Figura 9-5.

3. *Costante di Tempo:* Il caricamento di un canale del CTC con una Costante di Tempo è la prima operazione da eseguire prima di farlo funzionare.
La Costante di Tempo può avere un valore qualsiasi compreso tra 1 e 256, dove in particolare 00H verrà interpretato come 256. In Figura 9-6 è riportato il formato del byte della Costante di Tempo. Il caricamento di una Costante di Tempo in un canale del CTC avviene in due fasi distinte: nella prima fase una Parola di Controllo del Canale è posta in uscita verso il canale stesso con il bit

REGISTRO DI CONTROLLO CANALE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
ABILITAZ. INTERRUZ.	MODO	÷ 16 o 256	FRONTE ATTIVO	TRIGGER	CARICA COSTANTE DI TEMPO	RESET	1

USATI SOLO IN MODO CONTA-TEMPI

Figura 9-5. Formato del registro di controllo canale.

REGISTRO COSTANTE DI TEMPO

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CT_7	CT_6	CT_5	CT_4	CT_3	CT_2	CT_1	CT_0

BIT PIU' SIGNIF. — BIT MENO SIGNIF.

Figura 9-6. Formato del registro costante di tempo.

D2, di Caricamento della Costante di Tempo, posto a 1. In un secondo tempo avviene il caricamento vero e proprio; nel Registro Costante di Tempo del canale scelto, viene pari pari caricato il byte immediatamente successivo posto in uscita verso il canale stesso.

Il dispositivo CTC e la logica di controllo interruzioni dello Z80

Come abbiamo già avuto occasione di accennare, il CTC può generare delle richieste di servizio interruzione alla CPU Z80. I due requisiti preliminari sono che le interruzioni del CTC siano abilitate e che nel Registro del Vettore di Interruzione del CTC sia caricato un vettore di interruzione corretto. Nel capitolo relativo al circuito PIO vi è stata presentata la nozione di catena di priorità delle interruzioni *daisy-chain*. Nel Corso degli esperimenti avete potuto verificare che la posizione "logica", cioè la priorità di una porta del PIO all'interno delle daisy-chain è determinata solo dal modo in cui i suoi piedini IEI ed IEO sono collegati agli analoghi piedini degli altri PIO. Anche il CTC (come del resto tutti i circuiti appartenenti alla famiglia di dispositivi "periferici" Z80) è dotato dei piedini IEI ed IEO. I quattro canali interni al CTC, potendo ognuno indipendentemente richiedere una interruzione, occuperanno quattro locazioni successive all'interno della struttura daisy-chain di interruzione dello Z80. In questa sezione ci proponiamo appunto, di esaminare in modo particolareggiato la catena delle priorità delle interruzioni daisy-chain dello Z80. Anche se per lo più il nostro esame sarà rivolto al CTC, come elemento all'interno di una daisy chain, è comunque chiaro che la maggioranza delle considerazioni che faremo saranno applicabili a tutti i circuiti periferici della famiglia Z80.
Il circuito di Figura 9-7 illustra come è realizzata la logica di controllo delle interruzioni all'interno di ogni dispositivo della famiglia Z80. I tre segnali che richiedono particolare attenzione sono IEI, IEO ed $\overline{INT}$. Facciamo subito alcune osservazioni riguardo a questo circuito.

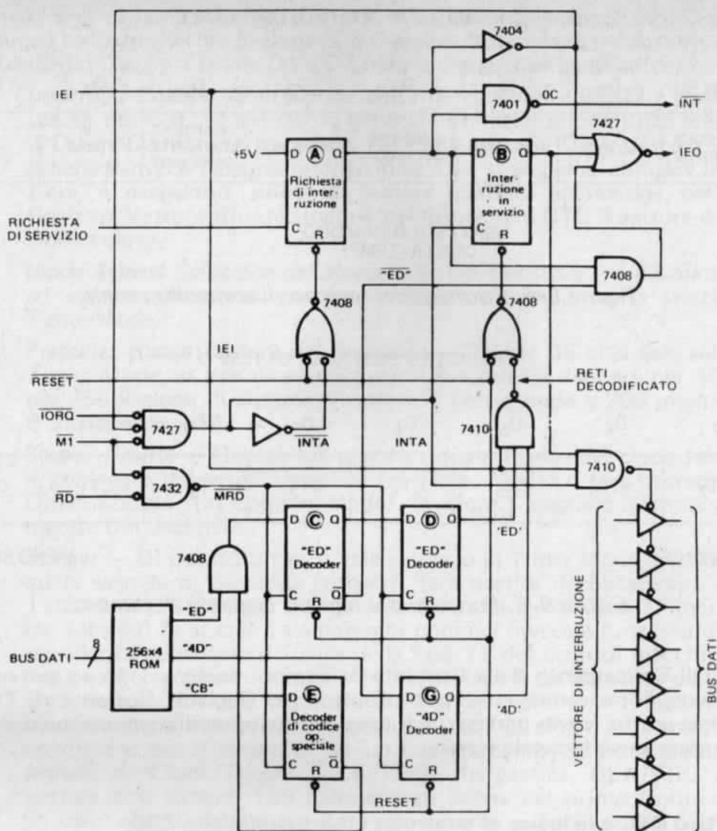


Figura 9-7. Logica di controllo delle interruzioni dello Z80.

- La richiesta di servizio (segnale Richiesta Servizio, NEED SERVICE), è determinata in base alle caratteristiche particolari del dispositivo periferico. Nel caso del CTC, l'impulso Richiesta Servizio è generato quando, ad interruzioni abilitate, il Down Counter di un canale giunge a zero.
- Un segnale Richiesta Servizio attivo ha come conseguenza che il Flip-Flop di Interruzione Pendente (Interrupt Pending) è posto a 1 (la sua uscita Q è, cioè posta a 1). Questo avviene *sempre*, indipendentemente dallo stato degli altri segnali.
- Se IEI è alto, un ciclo di riconoscimento dell'interruzione ($\overline{M1}$ e $\overline{IORQ}$ attivi) fa sì che il Flip-Flop di Interruzione Pendente sia posto a 0 e sia invece posto a 1 il Flip-Flop di Interruzione Sotto Servizio (Under Service, cioè, il cui servizio è in corso). Il fatto che IEI debba essere alto garantisce che il ciclo di riconoscimento dell'interruzione (Interrupt Acknowledge), vada ad interessare soltanto il dispositivo la cui interruzione deve essere servita.

- Il codice di identificazione (cioè il vettore di interruzione) del dispositivo la cui interruzione deve essere servita è posto sul bus dei dati nel corso del ciclo di riconoscimento dell'interruzione. Al fine di garantire che l'unico dispositivo a porre il suo vettore di interruzione sul bus dei dati sia quello sotto servizio, i buffer del bus del dispositivo sono abilitati se e soltanto se:
  1. Il Flip-Flop di Interruzione Sotto Servizio del dispositivo è posto a 1.
  2. L'ingresso IEI del dispositivo è alto.
  3. Il segnale di riconoscimento dell'interruzione è attivo.
- Se IEI è basso il dispositivo NON può attivare $\overline{INT}$.
- Perchè il dispositivo possa attivare $\overline{INT}$, devono del resto essere verificate le condizioni seguenti:
  1. IEI è alto
  2. Il Flip-Flop di Interruzione Pendente è posto a 1.
  3. Il Flip-Flop di Interruzione Sotto Servizio è posto a 0.

  Come diretta conseguenza di quest'ultimo punto secondo quanto abbiamo già osservato sperimentalmente nel corso delle esercitazioni sul PIO, un dispositivo (nel nostro caso un canale) non può interrompere "se stesso". Inoltre un dispositivo deve avere una richiesta di interruzione pendente anche se in attesa da un tempo brevissimo, prima di poter richiedere effettivamente l'interruzione della CPU.
- Affinchè IEO sia alto, devono essere soddisfatte TUTTE le condizioni seguenti:
  1. IEI deve essere alto.
  2. Il Flip-Flop di Interruzione Sotto Servizio deve essere posto a 0.
  3. Il Flip-Flop di Interruzione Pendente deve essere posto a 0.

  OPPURE

  Sul bus dei dati deve essere presente il codice operativo ED per tutto un ciclo $\overline{M1}$.
- Le linee dei dati sono tenute costantemente sotto controllo alla ricerca di una istruzione RETI (Ritorno dall'Interruzione, ED 4D esadecimale).

Per approfondire quanto abbiamo appena detto, prendiamo in considerazione due esempi:

Esempio 1: Le interruzioni dello Z80 non sono mai disabilitate.
Si esamini la struttura daisy-chain con quattro dispositivi (canali) illustrata in Figura 9-8. Nel primo schema (Figura 9-8A), i segnali IEI ed IEO sono tutti alti, il che significa che non vi è alcuna interruzione pendente. Il dispositivo a priorità più elevata (Canale 0) ha il proprio ingresso IEI collegato a +5 volt. Il servizio quindi, di una interruzione del Canale 0 non può mai essere sospeso in seguito ad una richiesta fatta dai Canali 1, 2 o 3, mentre, viceversa, una richiesta di interruzione del canale 0 potrà far sospendere il servizio di una interruzione dei Canali 1, 2 o 3, sempre che le interruzioni dello Z80 siano abilitate. Analizziamo una sequenza di eventi tipica:

1. Il Canale 2 inoltra una richiesta di interruzione che viene riconosciuta. La situazione che ne risulta è illustrata in Figura 9-8B. L'ingresso IEI del Canale 2 è alto (HI) mentre la sua uscita IEO è bassa (LO) e forza quindi basso l'ingresso IEI del Canale 3. Il Canale 3, essendo il suo ingresso IEI basso, porta nello stato basso la propria uscita IEO. Per gli altri dispositivi appartenenti alla daisy chain "a valle" del Canale 3, le linee IEI ed IEO sono forzate basse in maniera analoga. La linea IEO, posta bassa dal Canale 2, "propaga" cioè il livello basso verso il fondo della catena; il dispositivo caratterizzato da un ingresso IEI alto ed un'uscita IEO bassa è uno solo, e precisamente quello che ha avanzato la richiesta di interruzione ricevendo una conferma (riconoscimento) di servizio. Tutte le linee IEI ed IEO che precedono tale dispositivo, sono alte, mentre sono basse tutte quelle che lo seguono. Si faccia l'ipotesi

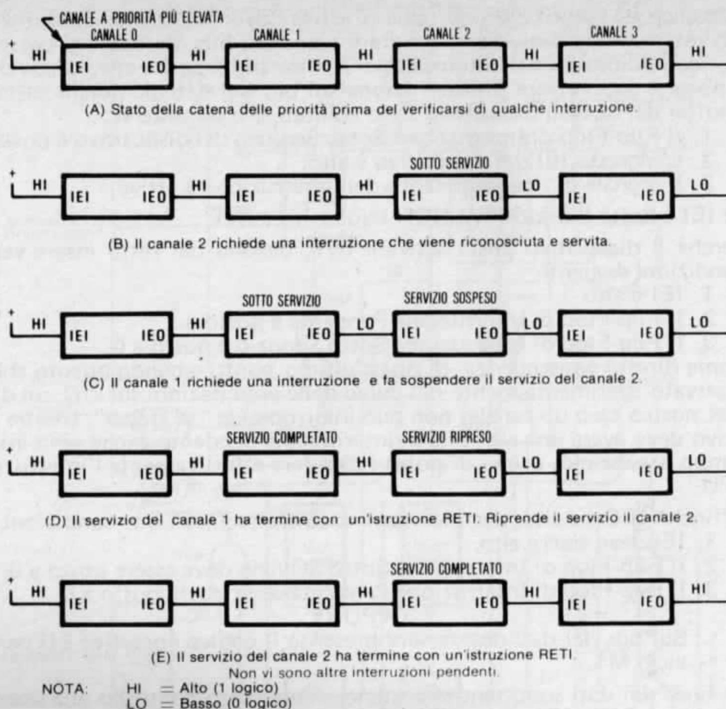


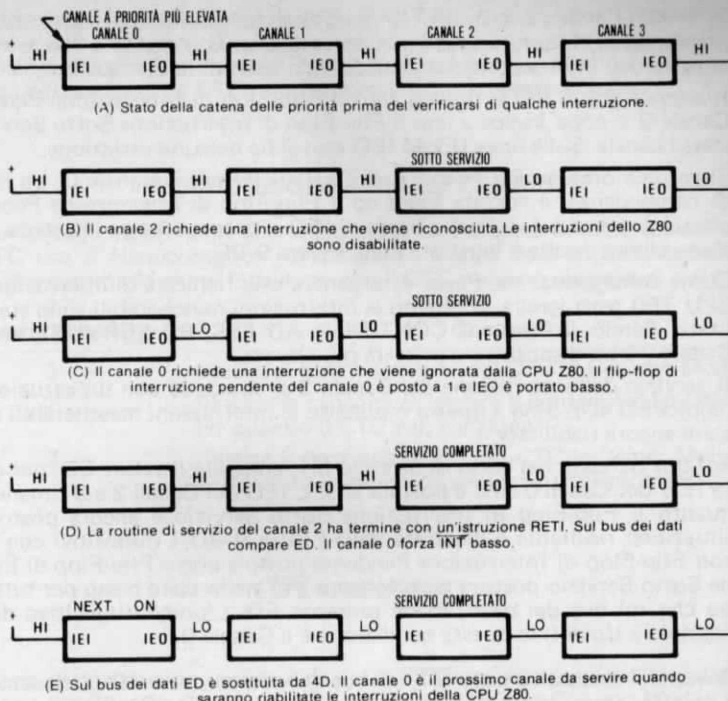
Figura 9-8. Catena di priorità delle interruzioni Daisy chain - Esempio 1.

che la routine di servizio dell'interruzione per il Canale 2 abiliti immediatamente le interruzioni dopo avere assunto il controllo della CPU. (Nell'Esempio 2 che seguirà formuleremo poi l'ipotesi inversa).

2. Il Canale 1 richiede una interruzione. Poichè la posizione che gli spetta nella catena a priorità è più elevata e poichè le interruzioni dello Z80 sono abilitate, il servizio del Canale 2 è momentaneamente sospeso. Avviene così il riconoscimento della richiesta di interruzione del Canale 1 ed ha inizio il servizio. La situazione che ne risulta è rappresentata dalla Figura 9-8C.

3. Viene completato il servizio dell'interruzione del Canale 1. Tutti i canali appartenenti alla catena "stanno ispezionando" il bus dei dati della CPU alla ricerca di un'istruzione RETI. Quando tale istruzione viene rilevata, ciascun canale aggiorna lo stato delle proprie linee IEO: la situazione risultante è illustrata nella Figura 9-8D. Poichè il servizio dell'interruzione relativo al Canale 2 era rimasto in attesa e, per il momento, nessun dispositivo a priorità più elevata sta avanzando una richiesta di servizio, la linea IEI del Canale 2 si porta alta, mentre rimane bassa la linea IEO. Riprende a questo punto il servizio dell'interruzione del Canale 2.

4. Il servizio dell'interruzione del Canale 2 si conclude con l'esecuzione di una nuova istruzione RETI. Ancora tutti i canali leggono l'istruzione RETI sul bus dei dati e aggiornano di conseguenza lo stato delle loro linee IEO. La situazione che ne risulta è rappresentata in Figura 9-8E.



(A) Stato della catena delle priorità prima del verificarsi di qualche interruzione.

(B) Il canale 2 richiede una interruzione che viene riconosciuta.Le interruzioni dello Z80 sono disabilitate.

(C) Il canale 0 richiede una interruzione che viene ignorata dalla CPU Z80. Il flip-flop di interruzione pendente del canale 0 è posto a 1 e IEO è portato basso.

(D) La routine di servizio del canale 2 ha termine con un'istruzione RETI. Sul bus dei dati compare ED. Il canale 0 forza $\overline{INT}$ basso.

(E) Sul bus dei dati ED è sostituita da 4D. Il canale 0 è il prossimo canale da servire quando saranno riabilitate le interruzioni della CPU Z80.

Figura 9-9. Catena di priorità delle interruzioni Daisy chain - Esempio 2.

Esempio 2: Le interruzioni dello Z80 non sono riabilitate dopo una interruzione iniziale.
Supponiamo di avere a che fare con una struttura a daisy-chain come quella di Figura 9-9A. Come nel caso precedente il Canale 0 rappresenta il dispositivo a priorità più elevata. Inizialmente non vi sono ne interruzioni sotto servizio né interruzioni pendenti. In questo esempio seguiamo più da vicino il circuito di Figura 9-7. In particolare rivolgiamo la nostra attenzione allo stato dei Flip-Flop di Interruzione Pendente e di Interruzione Sotto Servizio. Analissiamo la sequenza di eventi:

1. Nel Canale 2, essendosi verificata la necessità di un servizio viene generata la successione di eventi seguente:
   a. L'ingresso di clock del Flip-Flop di Interruzione Pendente riceve un impulso ed il Flip-Flop è, quindi, posto a 1.
   b. Poichè la linea IEI del Canale 2 si trova nello stato alto, il Flip-Flop di Interruzione Sotto Servizio è posto a zero ed il Flip-Flop di Interruzione Pendente è posto a uno, si verifica l'attivazione di $\overline{INT}$.
   c. La linea IEO del Canale 2 è forzata bassa. Tale livello si propaga lungo la linea dei dispositivi a priorità inferiore presenti nella daisy-chain. Si veda la Figura 9-9B.

   Viene così generata una richiesta di interruzione (Interrupt Request).
2. La CPU Z80 sente il livello basso sulla linea $\overline{INT}$, controlla il proprio Flip-Flop di Interruzione IFF1 (che supponiamo già abilitato) ed esegue un ciclo di riconoscimento dell'interruzione. Prima di servire l'interruzione del Canale

2, la CPU pone a zero IFF1, disabilitando quindi successive interruzioni mascherabili. L'ipotesi che noi facciamo a questo punto è che la routine di servizio dell'interruzione del Canale 2 *non* riabiliti le interruzioni.

3. Il segnale attivo $\overline{INTA}$ pone a zero il Flip-Flop di Interruzione Pendente del Canale 2 e pone invece a uno il Flip-Flop di Interruzione Sotto Servizio dello stesso canale. Sulle linee IEI ed IEO non si ha nessuna variazione.
4. Si verifica ora una RICHIESTA di SERVIZIO per il Canale 0. La linea IEO, di conseguenza, è forzata bassa ed il Flip-Flop di Interruzione Pendente (in attesa) è posto a 1. Il livello basso di IEO si propaga lungo la catena. La situazione che ne risulta è illustrata dalla Figura 9-9C.
5. Come conseguenza del Passo 4, si genera una richiesta di interruzione, che la CPU Z80 però ignora in quanto le interruzioni mascherabili sono state disabilitate. Perciò il Canale 2 ÇONTINUA AD ESSERE SERVITO, anche se il Canale 0 è un dispositivo a priorità più elevata.
6. Il servizio dell'interruzione del Canale 2 si conclude con un'istruzione RETI (codice ED 4D). Sino a questo momento le interruzioni mascherabili non sono state ancora riabilitate.
7. Sul bus dei dati, nel corso di un ciclo $\overline{M1}$, compare il valore ED, per cui l'uscita IEO dei Canali 0 ed 1 è portata alta. L'IEO dei Canali 2 e 3 rimane bassa in quanto il Flip-Flop di Interruzione Sotto Servizio è ancora posto a 1. La situazione risultante è illustrata nella Figura 9-9D. I dispositivi con IEI alto, con Flip-Flop di Interruzione Pendente posto a uno e Flip-Flop di Interruzione Sotto Servizio posto a zero forzano $\overline{INT}$ nello stato basso per tutto il tempo che sul bus dei dati rimane presente ED. L'unico dispositivo del nostro esempio a trovarsi in questa condizione è il Canale 0.
8. Durante il successivo ciclo $\overline{M1}$, sul bus dei dati compare 4D (l'istruzione RETI è caratterizzata dal codice operativo di due byte ED 4D). Perchè un dispositivo della catena che abbia rilevato il codice 4D possa porre a zero il proprio Flip-Flop di Interruzione Sotto Servizio, occorre che sia alto il suo ingresso IEI. Come si può vedere in Figura 9-9D, l'unico dispositivo della catena che abbia il piedino IEI alto ed il Flip-Flop di Interruzione Sotto Servizio posto a 1 è il dispositivo la cui interruzione è in quel momento sotto servizio, e precisamente il Canale 2. Per cui il Flip-Flop di Interruzione Sotto Servizio relativo al Canale 2 viene posto a zero.
9. Non appena dal bus dei dati scompare il segnale ED, tutti i dispositivi che hanno delle interruzioni pendenti portano la loro uscita IEO nello stato basso. Il canale 0, perciò porta IEO basso e questo segnale si propaga in avanti lungo la catena, secondo la situazione illustrata dalla Figura 9-9E.
10. La situazione di Figura 9-9E persiste sino a che non sono riabilitate le interruzioni mascherabili dello Z80.
11. Si faccia l'ipotesi che le interruzioni siano abilitate, appena prima dell'esecuzione dell'istruzione RETI, nella routine di servizio dell'interruzione del Canale 2. In tal caso, quando sul bus dei dati ED è scomparso e si è presentato 4D, tutti i dispositivi i cui Flip-Flop di Interruzione Pendente sono posti a uno portano immediatamente nello stato basso le rispettive linee IEO. Perciò, durante il successivo ciclo di riconoscimento dell'interruzione, l'unico dispositivo cui è consentito di porre il proprio vettore di interruzione sul bus dei dati è quello il cui IEI è alto, cioè quello a priorità più elevata.

Queste informazioni si possono trovare dettagliate nel *Manuale Tecnico sul CTC Z80* pubblicato a cura della Zilog e della SGS-ATES in cui è data una descrizione concisa, scritta con chiarezza ed organicità, di tutte le caratteristiche operative e funzionali

del dispositivo CTC. Se pensate di svolgere un lavoro approfondito con il CTC, tale manuale è quasi d'obbligo. Gli esperimenti che seguono sono così concepiti in modo da essere complemento al Manuale Tecnico e descrivono infatti un certo numero di esempi di utilizzazione del CTC non compresi nella letteratura citata.

INTRODUZIONE AGLI ESPERIMENTI

Nel corso degli esperimenti che seguono, vi sarà chiesto di interfacciare un dispositivo CTC con il Nanocomputer, utilizzando i segnali riportati sulla breadboarding station. Gli esperimenti che effettuerete con il CTC possono essere così riassunti:

Esperimento N.	Commenti
1	Illustra il funzionamento del CTC in Counter Mode.
2	Illustra la capacità di leggere il contenuto del Down Counter quando il CTC è in funzione.
3	Illustra il funzionamento del CTC in Timer Mode. Il CTC è utilizzato in modo da simulare un cronometro.

ESPERIMENTO N. 1

Scopo

Lo scopo di questo esperimento è quello di illustrare praticamente il funzionamento del Canale 1 del Circuito Contaeventi/Contatempi (CTC) in Counter Mode.

Schema del circuito (Figura 9-10)

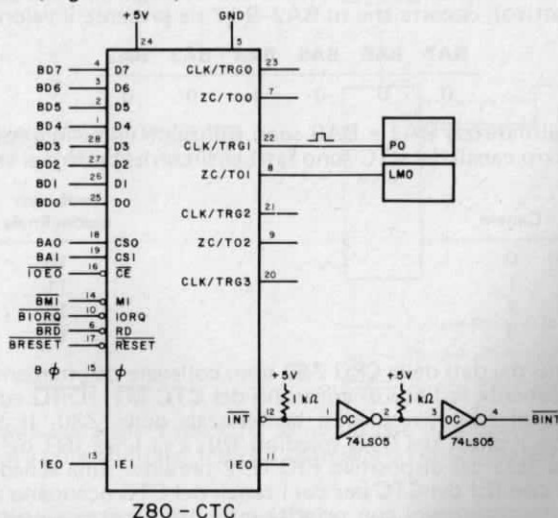


Figura 9-10. Schema N. 1. Circuito per l'Esperimento N. 1. Flusso del controllo tra INITC1, MAIN e SERV1.

Programmi MAIN, SERV1 e INITC1

Oltre alle routine MAIN e SERV1, per la cui descrizione si rimanda ai capitoli precedenti, in questo esperimento ci si vale del seguente programma:

Codice oggetto		Codice sorgente	Commenti
ED5E	INITC1:	IM2	; Modo 2 di interruzione dello Z80
21000F		LD HL,TABLE	; indirizzo della tabella dei vettori
7C		LD A,H	; byte più significativo dell'indirizzo
ED47		LD I,A	; scrivilo nel registro delle interruzioni
FD216E02		LD IY,SERV1	; indirizzo delle routine di servizio
FD221A0F		LD (TABLE+1AH),IY	; inseriscilo nella tabella
3E18		LD A,18H	; carica nel CANALE 0 del CTC
D310		OUT (10H),A	; il vettore di interruzione
08		EX AF,AF'	; predisposizione del formato per CONVDI
3E40		LD A,40H	
08		EX AF,AF'	
3EC7		LD A,0C7H	; predisponi la parola di controllo canale
D311		OUT (11H),A	
3E05		LD A,05H	; predisponi il registro
D311		OUT (11H),A	; della costante di tempo
C3C302		JP MAIN	; salta alla routine MAIN

Passo 1

Eseguite i collegamenti del circuito relativo a questo esperimento seguendo lo schema riportato in Figura 9-10. Si osservi come la decodifica degli indirizzi di questo circuito sia del tutto analoga a quella del chip PIO. Due linee, CS0 e CS1, sono collegate alle linee degli indirizzi BA0 e BA1. L'ingresso $\overline{CE}$ è collegato al segnale $\overline{IOE0}$ che è formato da una decodifica delle linee degli indirizzi BA2-BA7. Perchè $\overline{IOE0}$ sia basso (attivo), occorre che su BA2-BA7 sia presente il valore

BA7	BA6	BA5	BA4	BA3	BA2
0	0	0	1	0	0

Le due linee degli indirizzi BA1 e BA0 sono utilizzate come ingressi per la selezione del canale. I quattro canali del CTC sono fatti così corrispondere ai seguenti indirizzi:

Canale	Indirizzo esadecimale
0	10
1	11
2	12
3	13

Le linee del bus dei dati della CPU Z80 sono collegate direttamente al bus dei dati del CTC. Analogamente le linee di controllo del CTC $\overline{M1}$, $\overline{IORQ}$ ed $\overline{RD}$ sono collegate direttamente ai rispettivi piedini bufferizzati dello Z80. Il clock dello Z80, Φ, è collegato con il clock del CTC (piedino 15), e la linea $\overline{INT}$ del CTC è collegata a $\overline{BINT}$. La linea IEO del dispositivo PIO N. 2 presente sulla scheda del Nanocomputer è collegata con IEI del CTC per cui i canali del CTC occupano nella daisy chain dello Z80, le quattro posizioni con priorità inferiore rispetto a quelle del PIO N. 2. La linea IEO del chip CTC è invece aperta, cioè non collegata. La linea CLK/TRG1 (per il Canale 1) è collegata al pulsante P0 e quella ZC/TO1 ad un indicatore lumino-

so. Prima di far funzionare il CTC assicuratevi che i piedini dell'alimentazione siano collegati correttamente.

Passo 2

Il software utilizzato in questo esperimento è formato da tre routine, INITC1, MAIN e SERV1. La descrizione particolareggiata di MAIN e SERV1 è già stata fatta nei precedenti capitoli. Qui basterà sottolineare che MAIN costituisce il programma principale e SERV1 la routine di servizio delle interruzioni alla quale è trasferito il controllo quando si ha una interruzione generata dal CTC. INITC1 costituisce la routine di inizializzazione relativa a questo esperimento.
Essa esegue le seguenti operazioni:

1. Predispone il Modo 2 di interruzione dello Z80.
2. Carica il Registro di Interruzione dello Z80 con il byte di ordine superiore di TABLE.
3. Carica l'indirizzo della routine di servizio delle interruzioni, SERV1, nella locazione appropriata della tabella dei vettori di interruzione posta in memoria.
4. Carica il vettore di interruzione all'indirizzo della porta corrispondente al Canale 0 del CTC. Il vettore di interruzione è sempre scritto nel Canale 0. I cinque bit di ordine più elevato sono definiti dal programma mentre i bit di ordine inferiore sono automaticamente determinati dal CTC.
5. Carica il registro A' con 40H esadecimale per la subroutine CONVDI.
6. Carica la Parola di Controllo Canale nel Canale 1 del CTC.
7. Trascrivere il valore della Costante di Tempo nel canale 1 del CTC.
8. Salta alla routine MAIN.

La Figura 9-11 riporta il diagramma relativo al flusso del controllo tra INITC1, SERV1 e MAIN.

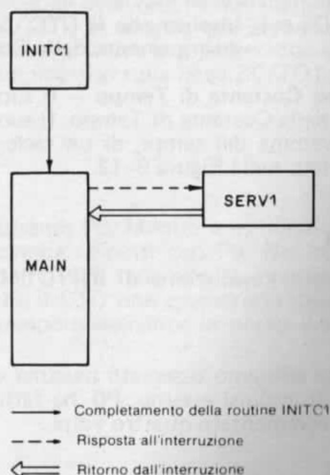


Figura 9-11. Flusso del controllo tra INITC1, MAIN e SERV1.

Per programmare il Canale 1 del CTC in modo che esso operi in Counter Mode e che abbia le interruzioni abilitate, INITC1 definisce il contenuto dei tre registri interni al chip CTC:

1. Registro Vettore di Interruzione del CTC
2. Registro Controllo Canale 1
3. Registro Costante di Tempo del Canale 1

Esaminiamo ora in particolare il modo in cui INITC1 programma il dispositivo CTC.

Registro Vettore di Interruzione – Prima di scegliere un vettore di interruzione conviene tenere conto della tabella dei vettori di interruzione già definiti negli esperimenti precedenti. E' così opportuno selezionare l'indirizzo nella tabella dei vettori di interruzione in modo che esso sia più grande di 0F11. Poichè il CTC specifica automaticamente il valore 0 in ciascuno dei tre bit meno significativi del vettore del Canale 0, la prima combinazione "disponibile" della tabella sarà 0F18H. INITC1 trascrive quindi il valore 18 nel Canale 0 del CTC, caricando così il byte di ordine inferiore del vettore di interruzione nel Registro Vettore di Interruzione del CTC. (Si osservi anche che INITC1 carica il registro I della CPU Z80 con 0F).

Registro di Controllo Canale – INITC1 pone il valore C7 in uscita verso la porta 11, il Canale 1 del CTC. Il CTC riconosce questo byte come una Parola di Controllo Canale a causa del bit D0 posto a uno. Il significato dei diversi bit di questa parola è il seguente:

Bit D7 = 1	Implica che sono abilitate le interruzioni del CTC. Ne consegue che, allorchè il Down Counter del Canale 1 giunge, nel suo conteggio, a zero, il CTC provvede all'attivazione della linea $\overline{\text{BINT}}$.
Bit D6 = 1	Implica che il Canale 1 deve funzionare in Counter Mode.
Bit D5	Non utilizzato in Counter Mode.
Bit D4 = 0	Implica che sia il fronte di discesa degli impulsi sulla linea CLK/TRG1 a decrementare il Down Counter.
Bit D3	Non utilizzato in Counter Mode.
Bit D2 = 1	Implica che il successivo byte inviato sarà il byte della Costante di Tempo.
Bit D1 = 1	Implica che il CTC, prima di cominciare il conteggio deve attendere il caricamento della Costante di Tempo.

Registro Costante di Tempo – Il successivo byte inviato da INITC1 alla porta 11 è quello della Costante di Tempo. Il suo valore è 05.
Il diagramma dei tempi, di un ciclo di scrittura, cioè di programmazione, del CTC è riportato nella Figura 9-12.

Passo 3

Iniziate l'esecuzione di INITC1. Date quattro impulsi con il pulsante P0. Cosa osservate?

Noi non abbiamo osservato nessuna variazione nella visualizzazione in corso. Il generatore di impulsi esterno, P0, ha fatto sì che il Down Counter del Canale 1 del CTC fosse decrementato quattro volte.

Passo 4

Inviate ancora un impulso mediante il pulsante, per un totale così di cinque impulsi. Che cosa osservate, a questo punto?

Noi abbiamo osservato che, non appena il CTC ha sentito il fronte di discesa del quinto impulso, esso ha generato una interruzione. La routine di servizio delle interruzioni SERV1 ha incrementato di 10 il solito contatore visualizzato ed ha poi restituito

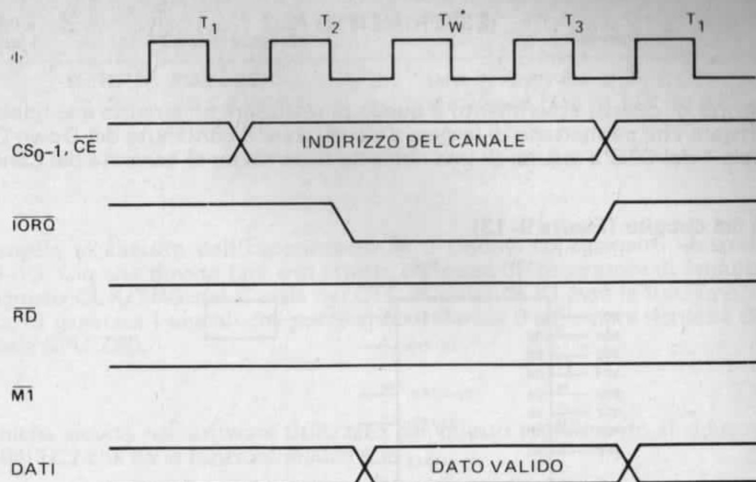


Figura 9-12. Ciclo di Scrittura nel CTC.

il controllo alla routine MAIN. Non ci è stato possibile osservare nessun cambiamento di stato sull'indicatore luminoso collegato a ZC/TO1, in quanto tale segnale è stato attivato per un tempo troppo breve perchè potessimo avere una percezione visiva di tale cambiamento. Per verificare la presenza di un impulso sulla linea ZC/TO1, potete collegare un flip-flop che lo memorizzi.

Passo 5

Si prema cinque volte (cinque impulsi) il pulsante P0. Mentre è in corso l'esecuzione di SERV1, si diano rapidamente altri cinque impulsi con P0. Noi abbiamo osservato che la prima interruzione generata dal CTC è stata servita, mentre per la seconda ciò non è avvenuto. Questo significa che il CTC non memorizza un'interruzione pendente per un canale di cui sia contemporaneamente in corso il servizio dell'interruzione precedente.

Passo 6

Si preme cinque volte il pulsante P0. Durante l'esecuzione di SERV1, generate rapidamente altri QUATTRO impulsi con P0. Una volta che SERV1 è stata completata, date ancora un impulso con P0. Noi abbiamo osservato che stavolta è stata generata un'interruzione. La nostra conclusione è che il Down Counter del Canale 1 del CTC continua ad operare anche quando è in corso il servizio dell'interruzione precedente; il CTC però non può in tale situazione memorizzare una interruzione pendente. *Non smontate il circuito utilizzato in questo esperimento in quanto esso sarà utilizzato per l'Esperimento N. 2.*

ESPERIMENTO N. 2

Scopo

Lo scopo di questo esperimento è quello di realizzare un circuito e le relative routine software che permettano di leggere e visualizzare il contenuto del Down Counter del Canale 1 del CTC a seguito di una richiesta (interruzione) generata dal Canale 0.

Schema del circuito (Figura 9-13)

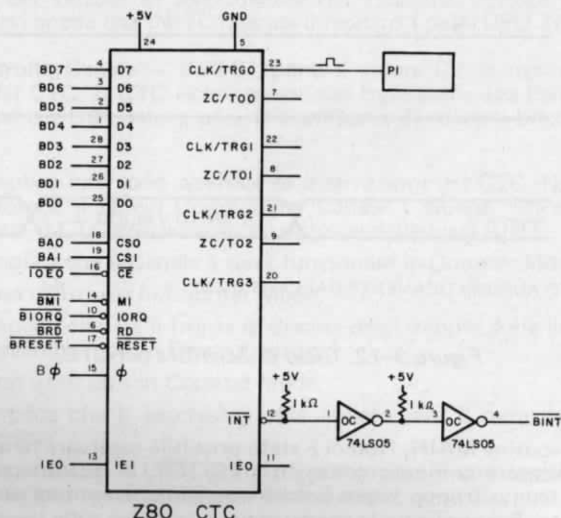


Figura 9-13. Schema N. 2 - Collegamenti addizionali per l'Esperimento N. 2 CTC.

Nota: Questo schema mostra solo le *aggiunte* che devono essere fatte allo schema dell'Esperimento N. 1, illustrato nella Figura 9-10.

Programmi MAIN, SERV1, INITC1, INITC2 e SERCT1

Nel corso di questo esperimento, oltre che delle routine MAIN, SERV1 e INITC1, ci si varrà dei programmi seguenti:

Codice oggetto		Codice sorgente	Commenti
FD21E006	INITC2:	LD IY,SERCT1	; indirizzo della routine di servizio
FD22180F		LD (TABLE+18H),IY	; inseriscilo nella tabella
3EC7		LD A,0C7H	; parola di controllo del canale 0
D310		OUT (10H),A	
3E01		LD A,01H	; registro della costante di tempo
D310		OUT (10H),A	; del canale 0
C3BD06		JP INITC1	

Codice oggetto		Codice sorgente	Commenti
C5	SERCT1:	PUSH BC	; salva i registri BC
0E11		LD C,11H	; seleziona la PORTA 11H del CTC
C33104		JP SERVI	

Passo 1

Aggiungete al circuito dell'Esperimento N. 1 i nuovi collegamenti illustrati nella Figura 9-13. Ciò che dovete fare è in effetti, collegare un generatore di impulsi esterno all'ingresso CLK/TRG del Canale del CTC. Il pulsante P1 avrà la funzione, nel nostro caso, di generare i segnali che porteranno il Canale 0 ad inviare richieste di interruzione alla CPU Z80.

Passo 2

Le uniche novità nel software utilizzato per questo esperimento si riducono alla routine INITC2 che ha le funzioni seguenti:

1. Carica la tabella dei vettori con l'indirizzo di SERCT1.
2. Carica con C7 esadecimale il Registro di Controllo Canale 0. (Si noti che si tratta della stessa parola usata per il Canale 1 sia nell'Esperimento N. 1 che in questo esperimento).
3. Carica con 01 esadecimale il Registro Costante di Tempo del Canale 0.
4. Salta a INITC1.

Riassumendo, il ruolo di INITC2 è quello di programmare il Canale 0 per il funzionamento in Counter Mode, impostando la Costante di Tempo minima, cioè 1. SERCT1 è una routine di servizio dell'interruzione funzionalmente identica a SERVID. La sua funzione principale è quella di leggere e visualizzare il contenuto di una determinata porta di ingresso. Per utilizzare SERCT1 è necessario apportare una modifica alla routine SERVI: nella locazione DSG+6H, sostituite l'istruzione NOP (00 esadecimale) con l'istruzione EI (FB esadecimale).
Tale sostituzione fà sì che le interruzioni vengano abilitate quando è ancora in corso l'esecuzione di SERCT1, evidenziando così meglio alcune proprietà del CTC. Siete pregati di apportare subito questa modifica.

Riassumiamo brevemente l'interazione tra gli elementi hardware e le varie parti software utilizzate per questo esperimento. Il pulsante P1 interagisce con il Canale 0 del CTC in modo da inviare una richiesta di interruzione ad ogni impulso. Tale interruzione è servita dalla routine SERCT1, che provvede a leggere e a visualizzare il contenuto del Down Counter del Canale 1. Quest'ultimo a sua volta viene modificato dagli impulsi generati dal pulsante P0. Cinque impulsi generati da P0 fanno sì che il Down Counter del Canale 1 raggiunga la condizione di zero — count (conteggio zero), con la conseguente attivazione di $\overline{\text{BINT}}$ da parte del Canale 1. La routine di servizio delle interruzioni per il Canale 1 è SERV1. Poichè più dispositivi possono richiedere una interruzione alla CPU (il Canale 0 e quello 1), si può creare una stiuazione di conflitto ed è necessario decidere a chi spetti la priorità nel ricevere il servizio. Cosa avviene se il Canale 1 avanza una richiesta di interruzione mentre è in corso il servizio di una interruzione del Canale 0? e viceversa? Le risposte a queste domande definiscono i passaggi, cioè il flusso del controllo possibile, tra le cinque routine utilizzate in questo esperimento. In Figura 9-14 sono riportati i possibili trasferimenti del controllo tra INITC1, INITC2, MAIN, SERCT1 e SERV1. Nei diversi passi di questo esperimento avrete modo di analizzare a fondo la struttura di priorità realizzata per i Canali del CTC.

In Figura 9-15 è riportato il diagramma dei tempi di un'operazione di lettura eseguita dalla routine SERCT1.

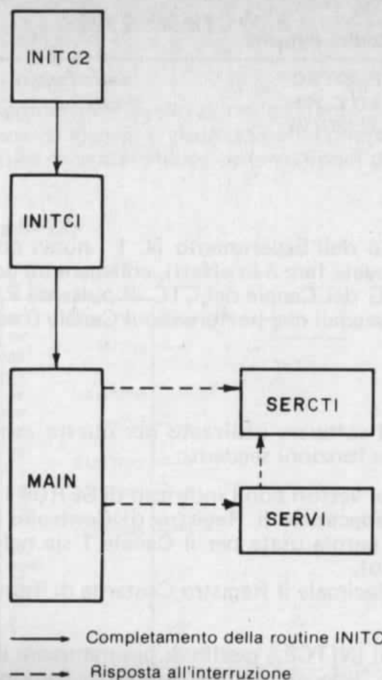


Figura 9-14. Flusso del controllo tra INITC1, INITC2, MAIN, SERCT1 e SERV1.

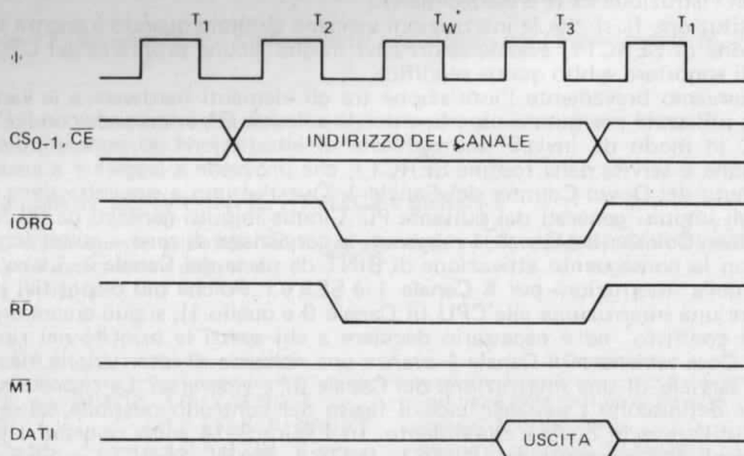


Figura 9-15. Ciclo di Lettura nel CTC.

Passo 3

Date inizio all'esecuzione della routine INITC2. Premete il pulsante P1. Cosa osservate?

Noi abbiamo osservato che per circa 10 secondi sull'unità tastiera/display del Nanocomputer veniva visualizzato il valore 05. Infatti il Down Counter del Canale 1 è stato caricato inizialmente con la Costante di Tempo del Canale 1 (05) e non vi è stato ancora nessun impulso sull'ingresso CLK/TRG1. In aggiunta abbiamo osservato che lo stack pointer è stato portato a 0EEC.

Passo 4

Premete il pulsante P0. Non dovreste notare nessun cambiamento del display. Premete ora P1. Cosa osservate?

Noi abbiamo osservato che sull'unità tastiera/display del Nanocomputer è stato visualizzato 04.

Passo 5

Premete tre volte di seguito P0. Non si dovrebbe notare nessuna variazione del display. Premete il pulsante P1. Dovreste osservare visualizzato 01 come valore attuale del Down Counter del Canale 1.

Passo 6

Azionate ancora una volta il pulser P0. Cosa osservate?

Noi abbiamo osservato che è stata generata una interruzione e che il controllo è passato alla routine SERV1. Lo stack pointer è stato portato a 0EEE.

Passo 7

Mentre SERV1 è in corso di esecuzione, date un impulso con P1. Cosa osservate?

Noi abbiamo osservato che l'esecuzione di SERV1 è stata sospesa al fine di servire la interruzione del Canale 0. Questo dimostra, tra l'altro, che il Canale 0 occupa nella catena una posizione di priorità più elevata di quella del Canale 1. Si noti che il valore visualizzato da SERCT1 per il Down Counter del Canale 1 è 5, confermando così il fatto che il contenuto del Registro Costante di Tempo è ricaricato immediatamente nel Down Counter, subito dopo il verificarsi di una condizione di conteggio zero.

Passo 8

Con lo stack pointer a 0F00, premete il pulsante P1. Mentre l'interruzione che ne risulta è sotto servizio, inviate rapidamente cinque impulsi con il pulsante P0. Che cosa osservate?

Noi abbiamo osservato che l'interruzione del canale 1 non viene servita immediatamente, ma che essa è stata lasciata in attesa (pendente) sino al completamente del

servizio dell'interruzione del Canale 0. Questo risultato conferma l'osservazione che abbiamo già fatto precedentemente a proposito della priorità maggiore del Canale 0 rispetto al Canale 1.

Nota: Anche se non l'avete completamente verificato in questi esperimenti, l'ordine di priorità (in ordine decrescente) dei canali del CTC è:

Canale 0 > Canale 1 > Canale 2 > Canale 3

ESPERIMENTO N. 3

Scopo

Lo scopo di questo esperimento è quello di illustrare praticamente il funzionamento in Timer Mode di un Canale del CTC. Programmando il Canale 0 per il Timer Mode e gli altri tre canali per il Counter Mode, potrete configurare il CTC per il suo funzionamento come cronometro.

Schema del circuito (Figura 9-16)

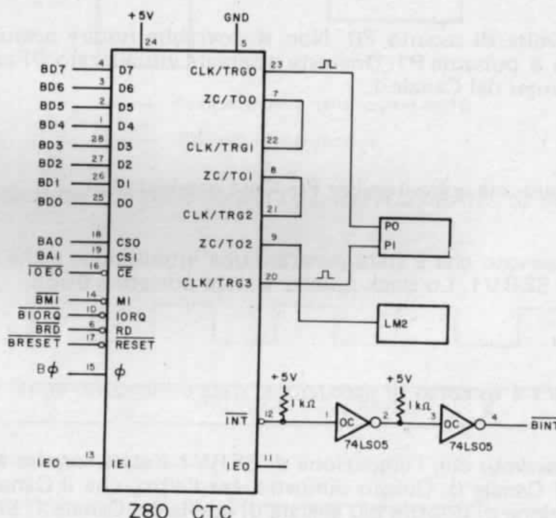


Figura 9-16. Schema N. 3 - Circuito per l'esperimento in Timer mode.

Programmi MAIN, INITC3 e SERCT2

In questo esperimento, oltre al programma MAIN, sono utilizzate le seguenti routine:

Codice oggetto		Codice sorgente	Commenti
ED5E	INITC3:	IM2	; Modo di interruzione 2 dello Z80
21000F		LD HL,TABLE	; tabella degli indirizzi dei vettori

Codice oggetto	Codice sorgente	Commenti
7C	LD A,H	; byte più significativo dell'indirizzo
ED47	LD I,A	; predispone il registro di interruzione
FD21E606	LD IY,SERCT2	; indirizzo della routine di servizio
FD22260F	LD (TABLE+26H),IY	; suo inserimento nella tabella
3E26	LD A,26H	; carica il vettore di interruzione
D314	OUT (14H),A	; nel canale 0 del CTC
08	EX AF,AF'	; predispone il formato per CONVDI
3E40	LD A,40H	
08	EX AF,AF'	
3E2F	LD A,2FH	; parola di controllo per il canale 0
D314	OUT (14H),A	
3E96	LD A,96H	; costante di tempo per il canale 0
D314	OUT (14H),A	
3E47	LD A,47H	; parola di controllo per il canale 1
D315	OUT (15H),A	
3E40	LD A,40H	; costante di tempo per il canale 1
D315	OUT (15H),A	
3E47	LD A,47H	; parola di controllo per il canale 2
D316	OUT (16H),A	
3E00	LD A,00H	; costante di tempo per il canale 2
D316	OUT (16H),A	
3EC7	LD A,0C7H	; parola di controllo per il canale 3
D317	OUT (17H),A	
3E01	LD A,01H	; costante di tempo per il canale 3
D317	OUT (17H),A	
C3C302	JP MAIN	

Codice oggetto	Codice sorgente	Commenti
C5	SERCT2: PUSH BC	; salva i registri della CPU
D5	PUSH DE	
E5	PUSH HL	
F5	PUSH AF	
DDE5	PUSH IX	
FDE5	PUSH IY	
FD2AE40F	LD IY,(ADDL)	; salva il contenuto di ADDL
FDE5	PUSH IY	
0E16	LD C,16H	; ingresso dal canale 2 del CTC
ED40	IN B,(C)	
AF	XOR A	; azzera A
90	SUB B	; cerca il numero di secondi
32E40F	LD (ADDL),A	; carica ADDL con il dato letto ; dal canale 2 del CTC
DD23	DST: INC IX	; aggiorna il puntatore dello stack dei dati
DD23	INC IX	
DD23	INC IX	
00	NOP	; nessuna operazione
DD3600FF	LD (IX+00H),0FFH	; predisposizione del tempo di DLOOPT
DD36010A	LD (IX+01H),00AH	; predisposizione del tempo di CLOOPT
DD360202	CLOOPT: LD (IX+02H),02H	; predisposizione del tempo di DLOOPT
21E50F	LD HL,ADDH	; punta al buffer del display

Codice oggetto		Codice sorgente	Commenti
ED57		LD A,I	; cerca il valore di IFF2
EA1C07		JP PE,HIGHT	
3600	LOWT:	LD (HL),00H	; valore = 0
1802		JR NEXT	
3610	HIGHT:	LD (HL),10H	; valore = 1
ED73E20F	NEXTT:	LD (DATAL),SP	; trascrivi SP nel buffer
21B90F		LD HL,LEDL	; predisponi HL per CONVDI
11E50F		LD DE,ADDH	; predisponi DE per CONVDI
CD7CFA		CAL CONVDI	
CD09F9	DLOOPT:	CALL DISPL	
DD3500		DEC (IX+00)	; temporizzatore per la visualizzazione
20F8		JR NZ,DLOOPT	
DD3502		DEC (IX+02)	; temporizzatore per la visualizzazione
20F3		JR NZ,DLOOPT	
DD3501		DEC (IX+01)	; temporizzatore per la routine di servizio
20CF		JR NZ,CLOOPT	
3E2F		LD A,2FH	; parola di controllo per il canale 0
D314		OUT (14H),A	
3E96		LD A,96H	; costante di tempo per il canale 0
D314		OUT (14H),A	
3E47		LD A,47H	; parola di controllo per il canale 1
D315		OUT (15H),A	
3E40		LD A,40H	; costante di tempo per il canale 1
D315		OUT (15H),A	
3E47		LD A,47H	; parola di controllo per il canale 2
D316		OUT (16H),A	
3E00		LD A,00H	; costante di tempo per il canale 2
D316		OUT (16H),A	
3EC7		LD A,0C7H	; parola di controllo per il canale 3
D317		OUT (17H),A	
3E01		LD A,01H	; costante di tempo per il canale 3
D317		OUT (17H),A	
FDE1		POP IY	; ripristina il contenuto di ADDL
FD22E40F		LD (ADDL),IY	
FDE1		POP IY	; ripristina i registri della CPU
DDE1		POP IX	
F1		POP AF	
E1		POP HL	
D1		POP DE	
C1		POP BC	
FB		EI	; abilita il flip-flop di interruzione
ED4D		RETI	; ritorna dalla interruzione

Passo 1

Effettuate i vari collegamenti del circuito secondo lo schema della Figura 9-16. Le differenze fondamentali tra questo circuito e quello utilizzato negli ultimi esperimenti sono due: in primo luogo, l'ingresso $\overline{\text{CE}}$, anzichè a $\overline{\text{IOEO}}$, è ora collegato con $\overline{\text{IOE1}}$. Per quanto riguarda gli indirizzi dei quattro canali del CTC, questo diverso collegamento darà luogo ai nuovi valori seguenti:

Canale	Indirizzo esadecimale della porta
0	14
1	15
2	16
3	17

In secondo luogo, la destinazione delle linee CLK/TRG e ZC/TO è cambiata. Si osservi che una condizione di Zero Count/Time out sul Canale 0 riporta un impulso sulla linea CLK/TRG1 del Canale 1. Se la medesima condizione di Zero Count/Time out si verifica sul Canale 1, sarà l'ingresso CLK/TRG2 del Canale 2 a ricevere un impulso della linea ZC/TO1. I canali 0, 1 e 2 si trovano, dunque, in CASCATA. La linea ZC/TO2 è collegata ad un indicatore luminoso. Al pulsante P0 fa capo la linea CLK/TRG0 del Canale 0 ed al pulsante P1 la linea CLK/TRG3 del Canale 3.

Passo 2

In questo esperimento, oltre che del programma MAIN, ci si vale delle routine INITC3 e SERCT2.

INITC3: Si tratta di una routine per l'inizializzazione che effettua le seguenti operazioni:

1. Imposta il Modo 2 di Interruzione dello Z80.
2. Posiziona la Tabella dei Vettori di Interruzione dello Z80 all'indirizzo TABLE.
3. Carica nella locazione TABLE+26H l'indirizzo della routine di servizio delle interruzioni SERCT2.
4. Carica con la locazione, nella Tabella dei Vettori di Interruzione, relativa all'indirizzo di SERCT2 il Registro Vettore di Interruzione del CTC.
5. Prepara il registro A' per CONVDI.
6. Carica il byte 2F nella Parola di Controllo del Canale 0. Tale byte è da interpretarsi come segue:
   - Bit D7 = 0 implica che le interruzioni del Canale 0 sono disabilitate.
   - Bit D6 = 0 implica che il Canale 0 funziona in Timer Mode.
   - Bit D5 = 1 implica che il fattore di Prescaler è 256.
   - Bit D4 = 0 implica che è il fronte di discesa del trigger a far partire il funzionamento del Temporizzatore.
   - Bit D3 = 1 implica che il Temporizzatore riceverà un impulso di trigger esterno, sulla linea CLK/TRG0.
   - Bit D2 = 1 implica che la prossima parola scritta nel Canale 0 sarà la Costante di Tempo.
   - Bit D1 = 1 implica che il Canale 0 deve riprendere ad operare non appena sia avvenuto il caricamento della Costante di Tempo.
   - Bit D0 = 1 implica che questo byte di dati rappresenta una Parola di Controllo Canale.
7. Carica la Costante di Tempo (96 esadecimale) del Canale 0.
8. Carica il byte 47 come Parola di Controllo del Canale 1.
   Questo byte fa assumere al Canale 1 una configurazione tale che esso funzioni esattamente nello stesso modo degli Esperimenti N. 1 e N. 2, con la differenza che, in questo caso, le interruzioni del CTC sono disabilitate. Il Canale funziona quindi, in Counter Mode, ma non sarà inviata alcuna richiesta di interruzione allo Z80 allorchè il

Down Counter giungerà ad un conteggio zero. La condizione di conteggio zero darà comunque luogo ad un impulso sulla linea ZC/TO1.

9. Carica la Costante di Tempo (40 esadecimale) del Canale 1.
10. Carica il byte 47 come Parola di Controllo del Canale 2, che, perciò, assumerà la stessa configurazione del Canale 1.
11. Carica la Costante di Tempo (00 esadecimale) del Canale 2.
    Per raggiungere la condizione di conteggio zero sarà dunque necessario contare sino a 256.
12. Carica il byte C7 come Parola di Controllo del Canale 3.
    Il Canale 3 assume, dunque, la medesima configurazione dei Canali 1 e 2, con la differenza che le interruzioni di tale canale sono abilitate.
13. Carica la Costante di Tempo (01 esadecimale) del Canale 3.

SERCT2: Si tratta della routine di servizio delle interruzioni generate dal Canale 3 del CTC allorchè si verifica la condizione di conteggio zero. Le funzioni eseguite da SERCT2 sono:

1. Salva nello stack il contenuto dei registri.
2. Salva lo stato del contatore visualizzato nel corso del programma MAIN.
3. Legge il valore del Down Counter del Canale 2.
4. Poichè ad ogni decremento, a partire da 00, del Down Counter del Canale 2 corrisponde un secondo di "tempo reale" trascorso, la differenza tra 00 ed il valore presente nel Down Counter è uguale al numero dei secondi trascorsi. Tale differenza è calcolata mediante le istruzioni:

```
XOR  A
SUB  B
```

5. Visualizza il numero dei secondi trascorsi nella posizione ADDL.
6. Visualizza il valore attuale di IFF2, il numero dei secondi trascorsi e il contenuto dello stack pointer.
7. Azzera il cronometro ricaricando di nuovo le Costanti di Tempo relative a ciascuno dei canali del CTC.
8. Ripristina i registri della CPU.
9. Abilita le interruzioni e ritorna (RETI).

Passo 3

In questo passo discutiamo i principi di funzionamento dell'hardware e software relativi all'esperimento. Facendo partire l'esecuzione dalla locazione INITC3, si ottiene la programmazione dei quattro canali del chip CTC nel modo sopra descritto. Il Canale 0 è configurato in modo da funzionare in Timer Mode ed inizia ad operare quando si presenta un impulso da P0. Premere P0 vuol dire, dunque, iniziare funzionamento del timer. Il Down Counter del Canale 0 raggiunge la condizione di conteggio zero un sessantaquattresimo di secondo più tardi ed invia un impulso sulla linea CLK/TRG1, facendo sì che il Down Counter del Canale 1 si decrementi di uno. Poichè il Canale 0 continua a generare un impulso sulla linea CLK/TRG1 ogni sessantaquattresimo di secondo, non è casuale che dopo sessantaquattro impulsi del genere, essendo trascorso in totale un secondo, il Down Counter del Canale 1 raggiunge la condizione di conteggio zero. Di conseguenza, sulla linea CLK/TRG2 si genera un impulso che fa decrementare di uno il Down Counter del Canale 2. Poichè questo Down Counter era stato inizializzato a 00, esso registra l'*opposto* dei secondi trascorsi, cioè zero meno il valore del Down Counter del Canale 2 è uguale al numero dei secondi che sono trascorsi. La routine di servizio delle interruzioni SERCT2 sfrutta questa circostanza per calcolare e visualizzare il tempo trascorso, misurato

in secondi. Si osservi come il massimo intervallo di tempo che questo contatempi ci consente di registrare è pari a 255 secondi.

Poichè il Canale 3 opera come contatore, con la Costante di Tempo uguale ad uno, poichè le sue interruzioni sono abilitate e poichè SERCT2 funge da routine di servizio per le sue interruzioni, un semplice azionamento del pulser P1 fa sì che sia visualizzato sull'unità tastiera/display del Nanocomputer, il tempo trascorso in secondi.

Passo 4

Provate il vostro temporizzatore dando inizio all'esecuzione di INITC3. Con l'ausilio di un normale orologio oppure, ancora meglio, di un cronometro, misurate il tempo che intercorre tra quando premete il pulsante P0 e quando premete P1. Noi abbiamo fatto numerose prove con diversi intervalli di tempo ed abbiamo scoperto che il cronometro basato sul CTC è davvero preciso. I risultati da voi ottenuti dovrebbero essere altrettanto lusinghieri!

[illegible]

1. [illegible] il Contatore di [illegible] del Canale 2 [illegible]

[illegible]

[illegible]

1. [illegible]
2. [illegible] MAIN
3. Leggi il valore del Green Counter del Canale 2
4. [illegible] Green Counter del Canale 2 [illegible]

[illegible]

[illegible]

[illegible]

APPENDICE A

DOCUMENTAZIONE DEL SOFTWARE UTILIZZATO NEL CORSO DEGLI ESPERIMENTI

Questa appendice contiene una documentazione addizionale, riguardante le routine utilizzate nel corso degli esperimenti. La prima parte del nostro studio verte sull'organizzazione generale della memoria. La Tavola A-1 illustra, per l'appunto, una mappa generale di memoria relativa ai 4K byte della memoria a lettura/scrittura.

Tavola A-1. Mappa di memoria relativa al software degli esperimenti con il Nanocomputer

0000 – 00FF	RAM pagina 0 (256 byte)
0100 – 07BC	Codice delle istruzioni dei programmi utilizzati nel corso degli esperimenti.
0800 – 0BFF	Lista REFILC per CHPTST
0C00 –	DSTACK valore iniziale del puntatore dello stack dei dati ↓ / lista UNKIC per CHPTST ↑
0EFF –	Limite dell'area usata dallo stack pointer dello Z80. Il valore iniziale di SP è 0F00 per il monitor del Nanocomputer e per tutte le routine utilizzate nel corso degli esperimenti ad eccezione di CHPTST. Quindi i dati sono salvati nelle locazioni a partire da 0F00 - 1 = 0EFF.
0F00 – 0F30	Tabella dei vettori di interruzione utilizzata dalle routine degli esperimenti ad eccezione di CHPTST ↑
0FA0 –	Valore iniziale dello stack pointer per CHPTST
0FAB – 0FFF	Locazioni di RAM utilizzate dal monitor del Nanocomputer e dalle routine per gli esperimenti.

Il blocco dei primi 256 byte della memoria a lettura/scrittura viene solitamente chimato Pagina 0. Essa è utilizzata dal set di istruzioni dello Z80, da due dei modi di interruzione dello Z80 e dal monitor del Nanocomputer. La Tavola A-2 mostra la sola Pagina 0 più in dettaglio.

In essa sono evidenziate tutte le 8 istruzioni di restart. Inoltre si ricorda che, la risposta ad un segnale attivo $\overline{INT}$ per il Modo 1 di Interruzione dello Z80 equivale, sotto l'aspetto funzionale, ad un restart alla locazione 0038, mentre la risposta ad un segnale attivo $\overline{NMI}$ consiste in un restart alla locazione 0066. Il monitor del Nanocomputer adopera la Pagina 0 per rientrare in se stesso allorchè è premuto il tasto RESET. Pertanto, volendo utilizzare la Pagina 0, è necessario procedere con la massima cautela.

Tavola A-2. Pagina 0 della RAM (256 byte)

	0000H–RST 00H
	0008H–RST 08H
	0010H–RST 10H
256 byte	0018H–RST 18H
	0020H–RST 20H
	0028H–RST 28H
	0030H–RST 30H
	0038H–RST 38H - Modo 1 di interruzione dello Z80
	0066H - Interruzione non mascherabile dello Z80
	00FFH

Il codice oggetto delle istruzioni eseguibili negli esperimenti è allocato nella memoria a partire dalla locazione 0100. La tabella dei Simboli Principale relativa a questo codice è riportata nella Tabella A-1. In essa a ciascun indirizzo simbolico è associato il corrispondente indirizzo assoluto di memoria. Analogamente la Tabella A-2 elenca i simboli e i relativi riscontri nei programmi. Tutti i simboli utilizzati nei programmi vi si trovano elencati in ordine alfabetico, con accanto, sulla stessa riga, il numero di linea delle istruzioni che fanno ad essi riferimento. Il listing completo dei vari programmi, con relativi numeri di linea, è raccolto nell'Appendice B. Poichè il software di cui ci si vale nei Capitoli da 1 a 5 è generalmente piuttosto semplice, il nostro esame è rivolto a quello della seconda metà di questo libro. L'area di memoria compresa tra INIT0 (021B) e DSTACK (DSTACK = 0C00 è il valore iniziale del puntatore dello stack dei dati), contiene *solamente* codice eseguibile e non può, quindi, essere utilizzato per la memorizzazione dei dati. La Tabella A-3 è una mappa particolareggiata di quest'area di memoria, dove le etichette si riferiscono ai punti di entrata del codice eseguibile. La corrispondenza tra le routine cui ci si riferisce nel corso degli esperimenti ed i punti di entrata del codice eseguibile è univoca.

L'analisi del modo di operare del puntatore dello stack dei dati è rimandata alle prossime pagine, tuttavia, è bene osservare subito che inizialmente esso punta alla cima del codice eseguibile (cioè alla sua più alta locazione di memoria), cosicchè lo stack dei dati "cresce in memoria verso l'alto" (cioè nel senso di indirizzi di memoria crescenti, da 0C00 in su).

Lo stack pointer del sistema dello Z80 è inizializzato dal monitor del Nanocomputer al valore 0F00. Di conseguenza i primi byte caricati nello stack andranno ad occupare le locazioni di memoria 0EFF e 0EFE. Poichè lo stack dello Z80 cresce in memoria verso il basso (si espande cioè progressivamente verso locazioni di memoria a indirizzo più basso), ne deriva che lo stack dei dati e quello del sistema crescono l'uno verso l'altro.

NOTA: Nella pratica è utile avere delle routine per il controllo di queste aree che si espandono dinamicamente. Per esempio, ogni volta che due byte sono caricati nello stack dello Z80, una routine apposita dovrebbe controllare che lo stack in questione non si stia sovrapponendo allo stack riservato ai dati; per contro, ogni qual volta che si procede ad un prelievo dallo

Tabella A-1. Tabella principale dei simboli

Simbolo	Indirizzo assoluto di memoria	Simbolo	Indirizzo assoluto di memoria	Simbolo	Indirizzo assoluto di memoria	Simbolo	Indirizzo assoluto di memoria	Simbolo	Indirizzo assoluto di memoria
ADD7	0FBA	ADDH	0FE5	ADDL	0FE4	BAD	06B9	BAUD	F9F2
BAUDRT	0FAE	BLKMVE	F000	CHECK	012F	CHECKB	F99D	CHPSTK	0FA0
CHPTST	0647	CLOOP1	0285	CLOOP3	0382	CLOOPG	0453	CLOOPM	0531
CLOOPN	0330	CLOOPT	070C	CLOOPX	05FE	COMPAR	0677	CONVDI	FA7C
CWORD	04F4	DATAH	0FE3	DATAL	0FE2	DATALP	01CA	DDRIVE	01B6
DECODE	010A	DELAY	01E3	DISAB	02E6	DISPL	F909	DISTST	01C7
DLOOP	02EA	DLOOP1	02A6	DLOOP3	03A4	DLOOPG	0472	DLOOPM	0550
DLOOPN	034F	DLOOPT	072B	DLOOPX	061F	DREGL	01E6	DS	F02D
DS1	0276	DS3	0373	DSG	0444	DSM	0522	DSN	0321
DSPLAY	0208	DST	06FD	DSTACK	0C00	DSX	05EF	ENABG	044A
END	0164	ENDREF	0669	ERRLP	015F	ERROR	0144	GETNO	01F3
GOOD	0686	ENPIO	03DD	HIGH1	0295	HIGH3	0392	HIGHG	0463
HIGHM	0541	HIGH	02D8	HIGHT	071C	HIGHX	060E	INITO	021B
INIT1	0231	HIGHN	0340	INIT2	0247	INITC1	06BD	INITC2	07A9
INITC3	076E	INIT1N	02F6	INITID	03F4	INITOC	03C1	INITPB	0495
INITPM	04D4	INITDC	05AA	KBSCAN	F8DB	KBTST	01EE	LEDH	0FB8
LEDL	0FB9	INITPP	0573	LOOP1	0100	LOOP2	0104	LOOP3	010E
LOOP4	0120	LENGHT	0700	LOOP6	01A0	LOW	02D4	LOW1	0291
LOW3	038E	LOOP5	019D	LOWM	053D	LOWN	033C	LOWT	0718
LOWX	060A	LOWG	045F	MASK	0694	MASKW	0003	MEM1	011E
MOVE	F014	MAIN	02C3	NEXT	02DA	NEXT1	0297	NEXT3	0394
NEXTB	067D	NANOR2	F00D	NEXTM	0543	NEXTN	0342	NEXTT	071E
NEXTX	0610	NEXTG	0465	NTEST	068B	NXTLOC	0135	NXTWD	06B3
OK	017F	NEXXT	013E	ORIGIN	0100	OUTPUT	01CD	OUTSIM	0212
OUTX	0637	ORGIN	0100	PSEL	0000	PULSR	0112	REF	065C
REFIC	0800	PCNTR	01AD	SERCT1	06E0	SERCT2	06E6	SEROCX	05E1
SERV1	026E	RESTAR	F08E	SERV3	036B	SERVI	0431	SERVIC	0419
SERVID	041F	SERV2	02F5	SERVIF	042B	SERVM	0505	SERVN	0319
SERVOC	03E8	SERVIE	0425	STORE	0688	STRING	F042	TABLE	0F00
TEST	0698	START	06BB	UCINM	01AB	UCINP	0190	UNKIC	0C00
UNKN	066A	THROW	03E3	XFER	0184				
		WAIT	019A						

Tabella A-2. Listing di tutti i riferimenti alle etichette

ADD7	0032	0367						
ADDH	0021	0153	0162	0171	0176	0366	0450	0460
		0484	0494	0540	0548	0582	0593	0714
		0722	0828	0836	0930	0940	1136	1144
ADDL	0020	0172	0638	0703	0706	0732	0810	0815
		0846	0921	0950	0951	1122	1128	1170
BAD	1079	1016						
BAUD	0037	0319						
BAUDRT	0029							
BLKMVE	1236							
CHECK	0123							
CHECKB	0030	0340	0372					
CHPSTK	0038	0983	0994					
CHPTST	0965							
CLOOP1	0449	0468						
CLOOP3	0581	0601						
CLOOPG	0713	0730						
CLOOPM	0827	0844						
CLOOPN	0539	0556						
CLOOPT	1135	1152						
CLOOPX	0929	0948						
COMPAR	1005							
CONVDI	0027	0163	0177	0368	0461	0496	0549	0594
		0723	0837	0941	1145			
CWORD	0792							
DATAH	0023	0160	0173					
DATAL	0024	0158	0174	0353	0458	0492	0546	0591
		0720	0834	0938	1142			
DATALP	0281	0314						
DDRIVE	0243							
DECODE	0065							
DELAY	0317	0300						
DISAB	0495							
DISPL	0028	0164	0178	0370	0462	0497	0550	0595
		0724	0838	0942	1146	1267		
DISTST	0277							
DLOOP	0497	0499						
DLOOP1	0462	0464	0466					
DLOOP3	0595	0597	0599					
DLOOPG	0724	0726	0728					
DLOOPM	0838	0840	0842					
DLOOPN	0550	0552	0554					
DLOOPT	1146	1148	1150					
DLOOPX	0942	0944	0946					
DREGL	0319	0333						
DS	1267	1269	1271					
DS1	0443							
DS3	0575							
DSG	0707							
DSM	0821							
DSN	0533							
DSPLAY	0370	0374						
DST	1129							

DSTACK	C036	0482	1249					
DSX	0923							
ENABG	0710							
END	0167	0142						
ENPIO	0627							
ENDREF	0992							
ERRLP	0164	0165						
ERROR	0145	0129						
GETNO	0345	0376						
GOOD	1016							
HIGH	0489	0486						
HIGH1	0455	0452						
HIGH3	0587	0584						
HIGHG	0719	0716						
HIGHM	0833	0830						
HIGHN	0545	0542						
HIGHT	1141	1138						
HIGHX	0935	0932						
INIT0	0393							
INIT1	0406							
INIT1N	0510							
INIT2	0419							
INITC1	1089	1221						
INITC2	1215							
INITC3	1183							
INITDC	0887							
INITID	0648							
INITOC	0614							
INITPB	0745							
INITPM	0777							
INITPP	0859	0910						
KBSCAN	0031	0345						
KBTST	0340	0342	0350					
LEDH	0025	1252	1264	1266				
LEDL	0026	0161	0175	0459	0493	0547	0592	0721
		0835	0939	1143				
LENGHT	1232	1240						
LOOP1	0046	0049	1311					
LOOP2	0055	0059						
LOOP3	0075	0079						
LOOP4	0109	0140						
LOOP5	0215	0223						
LOOP6	0216	0219						
LOW	0487							
LOW1	0453							
LOW3	0585							
LOWG	0717							
LOWM	0831							
LOWN	0543							
LOWT	1139							
LOWX	0933							
MAIN	0481	0401	0414	0432	0500	0522	0631	0664
		0772	0799	0882	1104	1210		
MASK	1039							
MASKW	0033	0969	1029	1051				
MEM1	0107	0184	0189					

MOVE	1252	1281						
NANOR2	1249							
NEXT	0490	0488						
NEXT1	0456	0454						
NEXT3	0588	0586						
NEXTB	1012	1020						
NEXTG	0720	0718						
NEXTM	0834	0832						
NEXTN	0546	0544						
NEXTT	1142	1140						
NEXTX	0936	0934						
NEXXT	0139	0131						
NTEST	1029	1071						
NXTLOC	0127	0135						
NXTWD	1067	1042						
OK	0178	0179	0184					
ORGIN	0017	0018						
ORIGIN	1231	1239						
OUTPUT	0286	0305						
OUTSIM	0382							
OUTX	0951							
PCNTR	0231	0237						
PSEL	0022	0251						
PULSR	0086							
REF	0983							
REFIC	0034	0985	1005					
RESTAR	1307	1237						
SERCT1	1109	1215						
SERCT2	1116	1187						
SEROCX	0915							
SERV1	0437	0395	0408	0423	0517	1093		
SERV2	0505	0425						
SERV3	0569	0427						
SERVI	0697	0671	0678	0685	0692	1111		
SERVIC	0669	0751	0863					
SERVID	0676	0652	0865					
SERVIE	0683	0891						
SERVIF	0690	0893						
SERVM	0804	0781						
SERVN	0527	0512						
SERVOC	0636	0618	0749					
START	1082	1022	1079					
STORE	1027	0990	1001					
STRING	1288	1251						
TABLE	0019	0089	0420	0424	0426	0428	0615	0619
		0649	0653	0746	0750	0752	0778	0782
		0860	0864	0866	0888	0892	0894	1090
		1094	1184	1188	1216			
TEST	1045							
THROW	0630							
UCINM	0229							
UCINP	0200							
UNKIC	0035	0996	1008					
UNKN	0994	1082						
WAIT	0214	0202	0235					
XFER	0184							

Tabella A-3. Punti di entrata nel software utilizzato per gli esperimenti con il Nanocomputer.

Punto di entrata	Nome della routine
0100	LOOP1
0104	LOOP2
010A	DECODE
0112	PULSR
011E	MEM1
0184	XFER
0190	UCINP
01AB	UCINM
01B6	DDRIVE
01C7	DISTST
01EE	KBTST
0212	OUTSIM
021B	INIT0
0231	INIT1
0247	INIT2
026E	SERV1
02C3	MAIN
02F5	SERV2
02F6	INIT1N
0319	SERVN
036B	SERV3
03C1	INITOC
03E8	SERVOC
03F4	INITID
0419	SERVIC
041F	SERVID
0425	SERVIE
042B	SERVIF
0431	SERVI
0495	INITPB
04D4	INITPM
0505	SERVM
0573	INITPP
05AA	INITDC
05E1	SEROCX
0647	CHPTST
06BD	INITC1
06E0	SERCT1
06E6	SERCT2
076E	INITC3
07A9	INITC2
F000	BLKMVE
F00D	NANOR2

stack dello Z80, un'altra routine dovrebbe verificare che lo stack non stia per "sfondare" oltre il suo limite 0F00.
Abbiamo rinunciato ad implementare in questa sede tali routine di gestione degli stack, in quanto si suppone che il lettore sia in grado di implementarle da sé senza difficoltà.

La tabella dei vettori è allocata a partire da 0F00. Una descrizione particolareggiata dei componenti la tabella dei vettori è riportata nella Tavola A-3.

Tavola A-3. Ultima pagina della RAM (software utilizzato per gli esperimenti con il Nanocomputer).

0EFF —	Stack Pointer dello Z80 (Il valore iniziale di SP è 0F00)	
0F00 —	SERV1	(Byte della parte meno significativa dell'indirizzo)
1	SERV1	(Byte della parte più significativa dell'indirizzo)
2	SERV2	
3	SERV2	
4	SERV3	
5	SERV3	
6	SERVOC	
7	SERVOC	
8	SERVID	
9	SERVID	
A	SERVIC	
B	SERVIC	
C	SERVM	
D	SERVM	
E	SERVIE	
F	SERVIE	
0F10	SERVIF	
11	SERVIF	
0F18	SERCT1	
0F19	SERCT1	
0F1A	SERV1	
0F1B	SERV1	
0F26	SERCT2	
0F27	SERCT2	
0FB9	LEDL	
0FE2	DATAL	
3	DATAH	
4	ADDL	
5	ADDH	
0FFF		

Per finire, gli ultimi (quelli ad indirizzo più alto) 84 byte della memoria a lettura/scrittura sono utilizzati dal monitor del Nanocomputer. Tra questi 84 byte ve ne sono 4 in particolare, ai quali le routine degli esperimenti fanno frequenti riferimenti. Le loro locazioni sono indicate nella Tavola A-3 (ADDH, ADDL, DATAH, DATAL) mentre la loro corrispondenza con le etichette che servono a individuarle e la sezione della tastiera dove il loro valore risulta visualizzato è illustrata in Figura A-1.

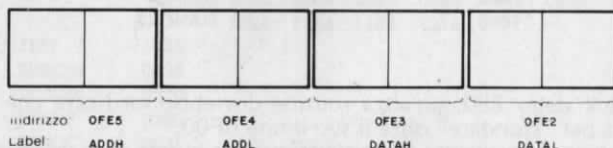


Figura A-1. Disposizione interna dei buffer utilizzati dal sistema operativo del Nanocomputer per pilotare il display sulla tastiera.

Nella Tabella A-4 sono riportati tutti gli 8 indirizzi relativi alle porte dei PIO; nelle routine che li useranno, i commenti faranno riferimento ai nomi delle porte corrispondenti.

Nel complesso, le routine utilizzate nei capitoli 6, 7 e 9 possono essere suddivise in tre classi principali:

CLASSE 1: Routine di inizializzazione.
INIT0, INIT1, INIT2, INIT1N, INIT0C, INITID, INITPB, INITPM, INITPP, INITDC, INITC1, INITC2, INITC3.

CLASSE 2: Routine di elaborazione principale.
MAIN

CLASSE 3: Routine di servizio delle interruzioni.
SERV1, SERV2, SERV3, SERVN, SERVOC, SERVID, SERVM, SERVIC, SERVIE, SERVIF, SERCT1, SERCT2, SEROCX, SERVI.

Tabella A-4. Tabella degli indirizzi dei PIO

PIO	Nome della porta	Tipo	Indirizzo della porta
PIO N. 2	PORTA C	DATI	08H
PIO N. 2	PORTA D	DATI	09H
PIO N. 2	PORTA C	CONTROLLO	0AH
PIO N. 2	PORTA D	CONTROLLO	0BH
PIO N. 3	PORTA E	DATI	0CH
PIO N. 3	PORTA F	DATI	0DH
PIO N. 3	PORTA E	CONTROLLO	0EH
PIO N. 3	PORTA F	CONTROLLO	0FH

Il PIO N. 2 è montato sul Nanocomputer (Q3) ed è accessibile grazie all'interfaccia sulla piastra di prova.

Il PIO N. 3 è un PIO cablato sullo zoccolo di montaggio dell'utente.

Routine della Classe 1

Le routine di inizializzazione non presentano particolari complicazioni. Forniremo qui ulteriori precisazioni soltanto per le seguenti istruzioni,

```
EX   AF, AF'
LD   A,40H
EX   AF,AF'
```

Questa sequenza di istruzioni predispone il formato per la visualizzazione di spazi vuoti (blank) per la routine CONVDI. Quest'ultima si vale del registro alternativo A' per stabilire quali degli otto display a sette segmenti della tastiera debbano accendersi e quali rimanere spenti. Se un display a sette segmenti deve essere acceso, nel corrispondente bit del registro A' deve esserci uno 0. La Figura A-2 illustra il formato degli 8 display normalmente utilizzato nel corso del libro; solamente il secondo elemento a sinistra del display è sempre lasciato spento. Il valore del registro A', utilizzato per generare questa configurazione del display è sempre 0 100 0000 ossia 40 (esadecimale). E' per tale motivo che questa sequenza di tre istruzioni è presente in tutte le routine di inizializzazione.

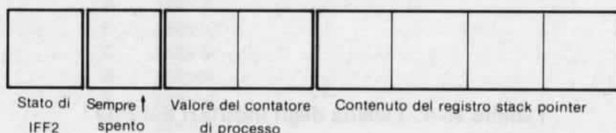


Figura A-2. Formato degli 8 caratteri del display della tastiera relativo al software utilizzato per gli esperimenti con il Nanocomputer.

Routine di Classe 2

Della routine MAIN sarà fornita una spiegazione particolareggiata, dato che molte delle funzioni da essa svolte sono analoghe a quelle di tutte le routine di servizio delle interruzioni.

La routine MAIN esegue 4 funzioni fondamentali.

1. Visualizza lo stato del flip-flop di interruzione, IFF2, sull'ultima cifra a sinistra del gruppo display/tastiera.
2. Visualizza il contenuto del registro stack pointer nelle prime quattro cifre di destra.
3. Decrementa ADDL, contatore di processo.
4. Visualizza ADDL sulla tastiera.

Per assolvere a queste funzioni, MAIN carica il buffer di 4 byte del display con i valori ottenuti in corrispondenza dell'ISTANTE DI CAMPIONAMENTO. Le Figure A-1 ed A-2 illustrano la corrispondenza tra gli indirizzi dei buffer dei display, le etichette utilizzate nel monitor del Nanocomputer per riferirsi ad essi e la posizione nel gruppo display assegnata a ciascuno di essi.

Esaminiamo ora MAIN in modo sequenziale e osserviamo come sono implementate le funzioni qui descritte. Abbiamo trascritto MAIN, corredandolo di un'ampia serie di commenti:

```
MAIN:   EI                  ; Abilita il flip-flop d'interruzione IFF1 dello Z80
        LD    IX,DSTACK     ; Inizializza il registro IX al valore DSTACK=0C00 che rap-
                            ; presenta il limite inferiore dello stack dei dati.
```

```
       LD    (IX+00),0FFH ; Inizializza a FFH il primo elemento dello stack dei dati.
                          ; Questo primo dato rappresenta il numero di volte che la rou-
                          ; tine DISPLAY dovrà essere chiamata per la visualizzazione
                          ; del valore presente in ADDL. Il valore FFH comporta una
                          ; visualizzazione dei dati per la durata di circa mezzo secondo.
       LD    HL,ADDH      ; Inizializza HL in modo da puntare al buffer che conterrà il
                          ; valore di IFF2.
       LD    A,I          ; Prende il valore di IFF2 e lo scrive nel buffer ADDH. Si
       JP    PE,HIGH      ; osservi come l'istruzione JP PE,HIGH costituisca il metodo
LOW:   LD    (HL),00H     ; più semplice per determinare e registrare il valore di IFF2.
       JR    NEXT         ;
HIGH:  LD    (HL),10H     ;
NEXT:  DEC   HL           ; Poni HL = ADDL facendo puntare HL al contatore di
                          ; processo
       DEC   (HL)         ; Decrementa il contatore di processo (ADDL).
       LD    (DATAL),SP   ; Metti lo stack pointer nel buffer, per poterlo visualizzare
                          ; in seguito
       LD    HL,LEDL      ; Predisponi HL e DE per la chiamata della subroutine
       LD    DE,ADDH      ; CONVDI
DISAB: NOP                ; Questo byte resta a disposizione per il futuro inserimento
                          ; di un'istruzione di un byte nel corso di un esperimento.
       CALL CONVDI        ; Con questa sequenza si visualizzano i parametri campionati.
DLOOP: CALL DISPLAY       ;
       DEC   (IX+00H)     ;
       JR    NZ,DLOOP     ;
                          ;
       JP    MAIN         ; Torna a campionare di nuovo i parametri.
```

Il registro IX è usato come una variabile globale, valida per tutte le routine relative ai vari esperimenti. Il registro IX, cioè, contiene sempre il valore corrente di ciò che noi intenderemo con il termine "puntatore dello stack dei dati", a prescindere dalla routine in corso di esecuzione. IX, perciò, è globale nel senso che tutte le routine lo utilizzano nello stesso identico modo e che esso riveste lo stesso ruolo in ciascuna routine. Una descrizione particolareggiata del modo di operare del puntatore dello stack dei dati è acclusa alla documentazione della routine SERV1. Per la comprensione della routine MAIN è necessario solo rendersi conto che LD IX,DSTACK inizializza a DSTACK il valore di questo puntatore dello stack dei dati.
Nel caso della routine MAIN è utilizzato un solo byte dello stack dei dati, (IX+00H) = (DSTACK+00H), ed esso contiene il valore della costante per la temporizzazione del loop di visualizzazione DLOOP. Tale costante di temporizzazione è inizializzata, come si è visto, ad FFH.

La sequenza di istruzioni:

```
       LD    HL,ADDH
       LD    A,I
       JP    PE,HIGH
LOW:   LD    (HL),00H
       JR    NEXT
HIGH:  LD    (HL),10H
NEXT:
```

è utilizzata per il campionamento del valore di IFF2 e per mettere a 0, oppure ad 1, il valore della cifra (in ADDH) che visualizza IFF2, a seconda che quest'ultimo sia 0 oppure 1. Questo risultato è ottenuto sfruttando il fatto che, all'esecuzione della istruzione LD A,I, lo stato di IFF2 è automaticamente trascritto nel flag di parità. Sappiamo altresì che la parità pari corrisponde a P = 1 e quella dispari a P = 0. Se,

dunque, P = 1, la parità è pari e la sequenza di istruzioni si svolge con il salto ad HIGH, mettendo a 0001 la porzione dei bit di maggior peso (D7-D4) di ADDH. Se P = 0, l'esecuzione dell'istruzione LD (HL), 10H è saltata e la parte superiore di ADDH, rimane uguale a 0. Si osservi che la parte superiore di ADDH corrisponda alla cifra visualizzata come IFF2.

La gestione dei contatori di processo avviene in base alla sequenza

```
DEC  HL
DEC  (HL)
```

Poichè HL è stato predisposto all'inizio in modo da puntare ad ADDH = 0FE5, DEC HL fa sì che la coppia di registri HL punti ora alla locazione 0EF4, corrispondente ad ADDL, buffer del contatore di processo.

LD (DATAL),SP prende lo stack pointer e lo carica in un buffer di memoria a due byte. Le istruzioni

```
LD   HL,LEDL
LD   DE,ADDH
```

predispongono i registri HL e DE per la subroutine CONVDI, che fa parte integrante del monitor del Nanocomputer. La visualizzazione è effettuata mediante esecuzioni ripetute della routine DISPLY del monitor del Nanocomputer.

Routine della Classe 3

La routine di servizio delle interruzioni SERV1 presenta molte caratteristiche tipiche del software in questa classe. Essa svolge le funzioni qui elencate:

1. Salva lo stato dei registri della CPU.
2. Aggiorna il puntatore dello stack dei dati.
3. Inizializza i buffer per la temporizzazione nello stack dei dati.
4. Campiona il valore del flip-flop di interruzione IFF2.
5. Gestisce il contatore di processo allocato in ADDL.
6. Campiona il valore dello stack pointer.
7. Oltre a visualizzare il valore presente nel contatore di processo, provvede anche alla visualizzazione dello stack pointer e di IFF2.
8. Gestisce un loop di temporizzazione che procederà al campionamento e relativa visualizzazione, ripetendosi 10 volte.
9. Ripristina lo stato dei registri della CPU.

Si osservi come le funzioni 4, 5, 6 e 7 di SERV1 siano essenzialmente le stesse che esegue anche MAIN. L'implementazione delle funzioni 1, 3, 8 e 9 appare priva di grosse complicazioni. Non rimane che da analizzare la funzione numero 2.

Passiamo ad illustrare cosa succede allo stack dei dati.

COMPORTAMENTO DEL PUNTATORE DELLO STACK DEI DATI

Il puntatore dello stack dei dati presenta nel suo modo di operare delle analogie con lo stack pointer dello Z80. Il fondo dello stack corrisponde al valore inizialmente assegnato al puntatore dello stack dei dati, e precisamente DSTACK. Questo stack "crescerà" in memoria verso l'alto, al contrario dello stack dello Z80 che cresce verso il basso. Il puntatore dello stack dei dati è inizializzato ad un valore tale per cui la crescita dello stack non rischi di scrivere sul codice eseguibile né su alcuna area riservata alle tabelle oppure ad altri buffer. La Mappa Principale di Memoria costituisce un quadro in cui è posta in rilievo la locazione DSTACK, fondo dello stack, rispetto alla locazione delle altre aree riservate ai buffer ed ai programmi.

Per lo stack dei dati non sono disponibili le istruzioni PUSH o POP. Questo stack, infatti, è manipolato in maniera lievemente diversa dall'altro. Ogni volta che si entra in una routine che dovrà usarlo, la stessa provvede a far avanzare di una posizione verso l'alto (un indirizzo di memoria) il puntatore dello stack dei dati per ciascun byte utilizzato. Ogni routine che sposta il puntatore dello stack dei dati deve provvedere al suo salvataggio prima di modificarlo ed al suo ripristino nel registro IX una volta che la sua esecuzione è giunta al completamento.

APPENDICE B

ELENCO DEGLI INDIRIZZI ASSOLUTI DELLE LOCAZIONI INDICATE CON NOTAZIONE SIMBOLICA NEL TESTO

Nome simbolico	Indirizzo assoluto
BAUDRT	0F AE
INMODE	0F AB
CONTST	FB 43
MEMTST	FA DC

LISTING COMPLETO DEI PROGRAMMI PER ESPERIMENTI

```
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                             NAME REL2.2
(0100)             ORGIN     EQU 0100H
                             ORG ORGIN
(0F00)             TABLE     EQU 0F00H
(0FE4)             ADDL      EQU 0FE4H
(0FE5)             ADDH      EQU 0FE5H
(0000)             PSEL      EQU 00H
(0FE3)             DATAH     EQU 0FE3H
(0FE2)             DATAL     EQU 0FE2H
(0FB8)             LEDH      EQU 0FB8H
(0FB9)             LEDL      EQU 0FB9H
(FA7C)             CONVDI    EQU 0FA7CH
(F909)             DISPL     EQU 0F909H
(0FAE)             BAUDRT    EQU 0FAEH
(F99D)             CHECKB    EQU 0F99DH
(F8DB)             KBSCAN    EQU 0F8DBH
```

```
(0FBA)                ADD7     EQU 0FBAH
(0003)                MASKW    EQU 0003H
(0800)                REFIC    EQU 0800H
(0C00)                UNKIC    EQU 0C00H
(0C00)                DSTACK   EQU 0C00H
(F9F2)                BAUD     EQU 0F9F2H
(0FA0)                CHPSTK   EQU 0FA0H
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                               NAME LOOP1
0100   D3C5           LOOP1:   OUT (0C5H),A     ;Output the contents
                                                ;of the accumulator
                                                ;to port C5
0102   18FC                    JR LOOP1         ;Repeat until break
                                                ;or reset
                      ;
                      ;
                      ;
                      ;
                               NAME LOOP2
0104   3E21           LOOP2:   LD A,21H         ;Initialize the ac-
                                                ;cumulator
0106   DBC5                    IN A,(0C5H)      ;Input a byte of
                                                ;data from port C5
0108   18FA                    JR LOOP2         ;Repeat until break
                                                ;or reset
                      ;
                      ;
                      ;
                               NAME DECODE
010A   0E20           DECODE:  LD C,20H         ;Load the device
                                                ;code into regis-
                                                ;ter C
010C   06C5                    LD B,0C5H        ;Load a nice look-
                                                ;ing byte into reg-
                                                ;ister B for sub-
                                                ;sequen. observa-
                                                ;tion on the upper
                                                ;half of the ad-
                                                ;dress bus
010E   ED61           LOOP3:   OUT (C),H        ;Output the content
                                                ;of the H register
                                                ;to port pointed to
                                                ;by register C
0110   18FC                    JR LOOP3         ;Repeat output in-
                                                ;struction until
                                                ;break or reset
                      ;
                      ;
                      ;
                               NAME PULSR
0112   0E20           PULSR:   LD C,20H         ;Load register C
```

```
                                        ;with the device
                                        ;code
0114    21000F           LD HL,TABLE    ;Load register
                                        ;pair HL with the
                                        ;starting memory
                                        ;address
0117    0608             LD B,08H       ;Load register B
                                        ;with the byte
                                        ;counter
0119    D3C0             OUT (0C0H),A   ;Clear the decade
                                        ;counter
011B    EDB3             OTIR           ;Output the byte
                                        ;string beginning
                                        ;at address HL of
                                        ;length (B) to port
                                        ;(C)
011D    76               HALT           ;Halt the CPU
                 ;
                 ;       NAME MEM1
                 ;
011E    3EFF     MEM1:   LD A,0FFH      ;Initialize the accu-
                                        ;mulator
0120    3C       LOOP4:  INC A          ;Begin memory test for
                                        ;next value
0121    32007F           LD (7F00H),A   ;Initialize location
                                        ;7F00 to contents of A
0124    01FF00           LD BC,00FFH    ;BC = byte counter for
                                        ;LDIR instruction
0127    11017F           LD DE,7F01H    ;DE = pointer to des-
                                        ;tination block
012A    21007F           LD HL,7F00H    ;HL = pointer to source
                                        ;block
012D    EDB0             LDIR           ;Load locations 7F00-
                                        ;7FFF with contents
                                        ;of register A
                 ;
012F    010001   CHECK:  LD BC,0100H    ;Check that above load
                                        ;worked, BC = byte cnt
0132    21007F           LD HL,7F00H    ;HL = pointer to loca-
                                        ;tion to be checked
0135    EDA1     NXTLOC: CPI            ;Compare (HL) with
                                        ;contents of A
0137    200B             JR NZ,ERROR    ;Mismatch indicates
                                        ;error
0139    E23E01           JP PO,NEXXT    ;Parity flag = 0 in-
                                        ;dicates BC = 0000,
                                        ;Go to next test
                                        ;byte (INC A)
013C    18F7             JR NXTLOC      ;Match and BC not =
                                        ;0000, go to next
                                        ;location
                 ;
013E    FEFF     NEXXT:  CP 0FFH        ;See if A = FF.
0140    20DE             JR NZ,LOOP4    ;If not, test
                                        ;next byte
```

```
0142   1820                JR END           ;If so, test
                                            ;is over
                  ;
0144   08         ERROR:   EX AF,AF'        ;Display error byte
                                            ;by using two rou-
                                            ;tines from Nanocom-
                                            ;puter operating
                                            ;system
0145   3E70                LD A,70H
0147   08                  EX AF,AF'
0148   3EE0                LD A,0E0H
014A   32E50F              LD (ADDH),A      ;Load 'E' in left-
                                            ;most display digit
014D   2B                  DEC HL           ;HL = pointer to bad
                                            ;location
014E   7D                  LD A,L
014F   32E20F              LD (DATAL),A
0152   7C                  LD A,H
0153   32E30F              LD (DATAH),A
0156   21B90F              LD HL,LEDL
0159   11E50F              LD DE,ADDH
015C   CD7CFA              CALL CONVDI
015F   CD09F9     ERRLP:   CALL DISPL
0162   18FB                JR ERRLP
                  ;
0164   08         END:     EX AF,AF'        ;Display F's if test OK
0165   3E00                LD A,00H
0167   08                  EX AF,AF'
0168   3EFF                LD A,0FFH
016A   32E50F              LD (ADDH),A
016D   32E40F              LD (ADDL),A
0170   32E30F              LD (DATAH),A
0173   32E20F              LD (DATAL),A
0176   21B90F              LD HL,LEDL
0179   11E50F              LD DE,ADDH
017C   CD7CFA              CALL CONVDI
017F   CD09F9     OK:      CALL DISPL
0182   18FB                JR OK
                  ;
                  ;
                  ;
                           NAME XFER
0184   016600     XFER:    LD BC,OK+5H—MEM1 ;Set-up for LDIR
                                            ;OK+5H—MEM1 is the
                                            ;number of bytes in pro-
                                            ;gram MEM1
0187   11007F              LD DE,7F00H      ;Destination is static RAM
018A   211E01              LD HL,MEM1       ;Source block is
                                            ;MEM1 program
018D   EDB0                LDIR             ;Do it
018F   FF                  RST 38H          ;Return control
                                            ;to the Nanocom-
                                            ;puter operating
                                            ;system
```

```
                 ;
                 ;
                 ;
                              NAME UCINP
0190  D311       UCINP:       OUT (11H),A     ;Latch data from
                                              ;logic switches
0192  CD9A01                  CALL WAIT       ;Delay for awhile
0195  0E12                    LD C,12H        ;Set up C register
                                              ;with input device
                                              ;code
0197  ED40                    IN B,(C)        ;Input data from
                                              ;latch into B regis-
                                              ;ter by enabling
                                              ;the buffers
0199  FF                      RST 38H         ;Return control to
                                              ;the Nanocomputer
                                              ;operating system
                 ;
019A  210500     WAIT:        LD HL,0005H     ;Delay loop
019D  11FFFF     LOOP5:       LD DE,0FFFFH
01A0  1B         LOOP6:       DEC DE
01A1  7A                      LD A,D
01A2  B3                      OR E
01A3  20FB                    JR NZ,LOOP6
01A5  2B                      DEC HL
01A6  7D                      LD A,L
01A7  B4                      OR H
01A8  20F3                    JR NZ,LOOP5
01AA  C9                      RET
                 ;
                 ;
                 ;
                              NAME UCINM
01AB  0E13       UCINM:       LD C,13H        ;Set up 13 as the
                                              ;device code
01AD  ED40       PCNTR:       IN B,(C)        ;Input pulse count
                                              ;to register B
01AF  ED41                    OUT (C),B       ;Output count to
                                              ;LEDs
01B1  CD9A01                  CALL WAIT       ;Delay before next
                                              ;count reading
01B4  18F7                    JR PCNTR        ;Repeat read/write/
                                              ;wait cycle
                 ;
                 ;
                 ;
                              NAME DDRIVE
01B6  010500     DDRIVE:      LD BC,0005H     ;B contains
                                              ;data to be
                                              ;displayed
                                              ;C contains
                                              ;device code
                                              ;for output
                                              ;port (PIO
                                              ;#1 B, data)
```

```
01B9  3E00                 LD A,PSEL       ;A contains
                                           ;the display
                                           ;position
                                           ;selector
01BB  00                   NOP             ;Filler so this
                                           ;program will
                                           ;fit inside of
                                           ;next program
                                           ;without having
                                           ;to reload most
                                           ;of the bytes
01BC  ED79                 OUT (C),A       ;Output display
                                           ;address to
                                           ;the HCF4514 by
                                           ;toggling bit
                                           ;D0
01BE  3C                   INC A
01BF  ED79                 OUT (C),A
01C1  3D                   DEC A
01C2  ED79                 OUT (C),A
01C4  ED41                 OUT (C),B       ;Output data
01C6  76                   HALT
                 ;
                 ;
                 ;
                           NAME DISTST
01C7  010500     DISTST:   LD BC,0005H     ;B contains data
                                           ;to be displayed
                                           ;C contains out-
                                           ;put device code
01CA  AF         DATALP:   XOR A           ;A contains the
                                           ;position to be
                                           ;displayed
01CB  160A                 LD D,0AH        ;D is the display
                                           ;position counter
01CD  ED79       OUTPUT:   OUT (C),A       ;Output display ad-
                                           ;dress to HCF4514
                                           ;by toggling bit D0
01CF  3C                   INC A
01D0  ED79                 OUT (C),A
01D2  3D                   DEC A
01D3  ED79                 OUT (C),A
01D5  ED41                 OUT (C),B       ;Output data
01D7  3C                   INC A           ;Increment position
                                           ;pointer to point to
                                           ;next display posi-
                                           ;tion
01D8  3C                   INC A
01D9  CDE301               CALL DELAY      ;Pause so display is
                                           ;constant for a short
                                           ;period
01DC  15                   DEC D           ;Decrement position
                                           ;counter
01DD  20EE                 JR NZ,OUTPUT    ;If D is not zero, then
```

```
                                    ;go back to output byte
                                    ;to next display posi-
                                    ;tion.
01DF  04                INC B       ;If all display posi-
                                    ;tions have been tested,
                                    ;update the output
                                    ;data
01E0  04                INC B
01E1  18E7              JR DATALP   ;Start again with new
                                    ;data byte
            ;
01E3  D5      DELAY:    PUSH DE     ;Save DE
01E4  16F0              LD D,0F0H   ;Timing byte
01E6  CDF2F9  DREGL:    CALL BAUD   ;BAUD is a routine
                                    ;in the operating
                                    ;system that delays
                                    ;exactly one sampling
                                    ;period. The
                                    ;length of the pe-
                                    ;riod is set via a
                                    ;timing byte stored
                                    ;in memory. In sub-
                                    ;routine DELAY, the
                                    ;delay will be 16
                                    ;(base 10) sampling
                                    ;periods
01E9  15                DEC D
01EA  20FA              JR NZ,DREGL
01EC  D1                POP DE      ;Restore DE
01ED  C9                RET
            ;
            ;
            ;
                        NAME KBTST
01EE  CD9DF9  KBTST:    CALL CHECKB ;Check for pressed
                                    ;key
01F1  28FB              JR Z,KBTST  ;Z-flag = 1 implies
                                    ;that no key is
                                    ;pressed
01F3  CDDBF8  GETNO:    CALL KBSCAN ;Z-flag = 0 implies
                                    ;that one or more
                                    ;keys are pressed.
                                    ;See if just one,
                                    ;and which one.
01F6  38F6              JR C,KBTST  ;C-flag = 1 implies
                                    ;that more than one
                                    ;key was pressed
01F8  32E20F            LD (DATAL),A ;C-flag = 0 implies
                                    ;that one key was
                                    ;pressed and its
                                    ;number is in reg-
                                    ;ister A. Display
                                    ;hex key number in
                                    ;data display posi-
                                    ;tions
```

```
01FB  08                  EX AF,AF'       ;Set up for call
                                          ;to CONDVI
01FC  3EFC                LD A,0FCH       ;Just display data
                                          ;digits
01FE  08                  EX AF,AF'
01FF  11E50F              LD DE,ADDH
0202  21B90F              LD HL,ADD7-1
0205  CD7CFA              CALL CONVDI     ;Translate key no
                                          ;for display
0208  CD09F9     DSPLAY:  CALL DISPL      ;Display the key
                                          ;number
020B  CD9DF9              CALL CHECKB     ;Check for pressed
                                          ;key
020E  28F8                JR Z,DSPLAY     ;Keep displaying
                                          ;if no key pressed
0210  18E1                JR GETNO        ;Get key number
                                          ;if key is pressed
                 ;
                 ;
                 ;
                          NAME OUTSIM
0212  3E0F       OUTSIM:  LD A,0FH        ;Program the PIO #2 to
                                          ;Mode 0
0214  D30A                OUT (0AH),A
0216  3E43                LD A,43H        ;Output the byte 43H
                                          ;to PC0-7 lines
0218  D308                OUT (08H),A
021A  76                  HALT
                 ;
                 ;
                 ;
                          NAME INIT0
021B  3EC3       INIT0:   LD A,0C3H       ;first byte is jump
021D  323800              LD (0038H),A    ;load into RST location
0220  FD216E02            LD IY,SERV1     ;address of service
0224  FD223900            LD (0039H),IY   ;routine #1
0228  ED46                IM0             ;Interrupt Mode 0
022A  08                  EX AF,AF'       ;set format for blanks
022B  3E40                LD A,40H        ;for CONVDI
022D  08                  EX AF,AF'
022E  C3C302              JP MAIN         ;Jump to routine MAIN
                 ;
                 ;
                 ;
                          NAME INIT1
0231  3EC3       INIT1:   LD A,0C3H       ;first byte is jump
0233  323800              LD (0038H),A
0236  FD216E02            LD IY,SERV1     ;address of service
023A  FD223900            LD (0039H),IY   ;routine #1
023E  ED56                IM1             ;Interrupt mode 1
0240  08                  EX AF,AF'       ;set format for blanks
0241  3E40                LD A,40H        ;for CONVDI
0243  08                  EX AF,AF'
0244  C3C302              JP MAIN         ;Jump to routine MAIN
                 ;
```

```
                    ;
                    ;
                                 NAME INIT2
0247  ED5E          INIT2:       IM2                  ;Interrupt mode 2
0249  21000F                     LD HL,TABLE          ;address of vector table
024C  7C                         LD A,H               ;high byte of address
024D  ED47                       LD I,A               ;set Interrupt register
024F  FD216E02                   LD IY,SERV1          ;first service routine
0253  FD22000F                   LD (TABLE),IY        ;set in vector table
0257  FD21F502                   LD IY,SERV2          ;second service routine
025B  FD22020F                   LD (TABLE+2),IY      ;set in vector table
025F  FD216B03                   LD IY,SERV3          ;third service routine
0263  FD22040F                   LD (TABLE+4),IY      ;set in vector table
0267  08                         EX AF,AF'            ;set format for CONVDI
0268  3E40                       LD A,40H
026A  08                         EX AF,AF'
026B  C3C302                     JP MAIN              ;Jump to routine MAIN
                    ;
                    ;
                    ;
                                 NAME SERV1
026E  C5            SERV1:       PUSH BC              ;save CPU registers
026F  D5                         PUSH DE
0270  E5                         PUSH HL
0271  F5                         PUSH AF
0272  DDE5                       PUSH IX
0274  FDE5                       PUSH IY
0276  DD23          DS1:         INC IX               ;update data stack pointer
0278  DD23                       INC IX
027A  DD23                       INC IX
027C  00                         NOP                  ;no operation
027D  DD3600FF                   LD (IX+00H),0FFH     ;set DLOOP1 time
0281  DD36010A                   LD (IX+01H),00AH     ;set CLOOP1 time
0285  DD360202      CLOOP1:      LD (IX+02H),02H      ;set DLOOP1 time
0289  21E50F                     LD HL,ADDH           ;point to display buffer
028C  ED57                       LD A,I               ;find value of IFF2
028E  EA9502                     JP PE,HIGH1
0291  3600          LOW1:        LD (HL),00H          ;value = 0
0293  1802                       JR NEXT1
0295  3610          HIGH1:       LD (HL),10H          ;value = 1
0297  2B            NEXT1:       DEC HL               ;move buffer pointer
0298  34                         INC (HL)             ;increment ADDL
0299  ED73E20F                   LD (DATAL),SP        ;copy SP to buffer
029D  21B90F                     LD HL,LEDL           ;set for CONVDI
02A0  11E50F                     LD DE,ADDH           ;set for CONVDI
02A3  CD7CFA                     CALL CONVDI
02A6  CD09F9        DLOOP1:      CALL DISPL
02A9  DD3500                     DEC (IX+00)          ;timer for display
02AC  20F8                       JR NZ,DLOOP1
02AE  DD3502                     DEC (IX+02)          ;timer for display
02B1  20F3                       JR NZ,DLOOP1
02B3  DD3501                     DEC (IX+01)          ;timer for service routine
02B6  20CD                       JR NZ,CLOOP1
02B8  FDE1                       POP IY               ;restore CPU registers
02BA  DDE1                       POP IX
```

```
02BC  F1                        POP AF
02BD  E1                        POP HL
02BE  D1                        POP DE
02BF  C1                        POP BC
02C0  FB                        EI                  ;enable interrupts
02C1  ED4D                      RETI                ;return from interrupt
                    ;
                    ;
                    ;
                                NAME MAIN
02C3  FB            MAIN:       EI                  ;enable interrupts
02C4  DD21000C                  LD IX,DSTACK        ;bottom of data stack
02C8  DD3600FF                  LD (IX+00H),0FFH    ;timer for display
02CC  21E50F                    LD HL,ADDH          ;set pointer to buffer
02CF  ED57                      LD A,I              ;find value of IFF2
02D1  EAD802                    JP PE,HIGH
02D4  3600          LOW:        LD (HL),00H         ;value = 0
02D6  1802                      JR NEXT
02D8  3610          HIGH:       LD (HL),10H         ;value = 1
02DA  2B            NEXT:       DEC HL              ;move buffer pointer
02DB  35                        DEC (HL)            ;decrement COUNT
02DC  ED73E20F                  LD (DATAL),SP       ;copy SP to buffer
02E0  21B90F                    LD HL,LEDL          ;set up for CONVDI
02E3  11E50F                    LD DE,ADDH          ;set up for CONVDI
02E6  00            DISAB:      NOP                 ;no operation
02E7  CD7CFA                    CALL CONVDI
02EA  CD09F9        DLOOP:      CALL DISPL
02ED  DD3500                    DEC (IX+00H)        ;timer for display
02F0  20F8                      JR NZ,DLOOP
02F2  C3C302                    JP MAIN             ;jump back to beginning
                    ;
                    ;
                    ;
                                NAME SERV2
02F5  76            SERV2:      HALT                ;Halt the microcomputer
                    ;
                    ;
                    ;
                                NAME INIT1N
02F6  3EC3          INIT1N:     LD A,0C3H           ;first byte is jump
02F8  326600                    LD (0066H),A        ;non-maskable interrupt
02FB  FD211903                  LD IY,SERVN         ;address of service for
02FF  FD226700                  LD (0067H),IY       ;non-maskable interrupt
0303  ED56                      IM1                 ;Interrupt mode 1
0305  3EC3                      LD A,0C3H           ;first byte is jump
0307  323800                    LD (0038H),A
030A  FD216E02                  LD IY,SERV1         ;address of service
030E  FD223900                  LD (0039H),IY       ;routine #1
0312  08                        EX AF,AF'           ;set format for blanks
0313  3E40                      LD A,40H            ;for CONVDI
0315  08                        EX AF,AF'
0316  C3C302                    JP MAIN             ;Jump to routine MAIN
                    ;
                    ;
                    ;
```

```
                          NAME SERVN
0319  C5        SERVN:    PUSH BC             ;save CPU registers
031A  D5                  PUSH DE
031B  E5                  PUSH HL
031C  F5                  PUSH AF
031D  DDE5                PUSH IX
031F  FDE5                PUSH IY
0321  DD23      DSN:      INC IX              ;update data stack pointer
0323  DD23                INC IX
0325  DD23                INC IX
0327  00                  NOP                 ;no operation
0328  DD3600FF            LD (IX+00H),0FFH    ;set DLOOPN time
032C  DD36010A            LD (IX+01H),00AH    ;set CLOOPN time
0330  DD360202  CLOOPN:   LD (IX+02H),02H     ;set DLOOPN time
0334  21E50F              LD HL,ADDH          ;point to display buffer
0337  ED57                LD A,I              ;find value of IFF2
0339  EA4003              JP PE,HIGHN
033C  3600      LOWN:     LD (HL),00H         ;value = 0
033E  1802                JR NEXTN
0340  3610      HIGHN:    LD (HL),10H         ;value = 1
0342  ED73E20F  NEXTN:    LD (DATAL),SP       ;copy SP to buffer
0346  21B90F              LD HL,LEDL          ;set for CONVDI
0349  11E50F              LD DE,ADDH          ;set for CONVDI
034C  CD7CFA              CALL CONVDI
034F  CD09F9    DLOOPN:   CALL DISPL
0352  DD3500              DEC (IX+00)         ;timer for display
0355  20F8                JR NZ,DLOOPN
0357  DD3502              DEC (IX+02)         ;timer for display
035A  20F3                JR NZ,DLOOPN
035C  DD3501              DEC (IX+01)         ;timer for service routine
035F  20CF                JR NZ,CLOOPN
0361  FDE1                POP IY              ;restore CPU registers
0363  DDE1                POP IX
0365  F1                  POP AF
0366  E1                  POP HL
0367  D1                  POP DE
0368  C1                  POP BC
0369  ED45                RETN                ;return from non-maskable
                                              ;interrupt
                ;
                ;
                ;
                          NAME SERV3
036B  C5        SERV3:    PUSH BC             ;save CPU registers
036C  D5                  PUSH DE
036D  E5                  PUSH HL
036E  F5                  PUSH AF
036F  DDE5                PUSH IX
0371  FDE5                PUSH IY
0373  DD23      DS3:      INC IX              ;update data stack pointer
0375  DD23                INC IX
0377  DD23                INC IX
0379  00                  NOP                 ;no operation
037A  DD3600FF            LD (IX+00H),0FFH    ;set DLOOP3 time
037E  DD36010A            LD (IX+01H),00AH    ;set CLOOP3 time
```

```
0382  DD360202          CLOOP3:  LD (IX+02H),02H    ;set DLOOP3 time
0386  21E50F                     LD HL,ADDH         ;point to display buffer
0389  ED57                       LD A,I             ;find value of IFF2
038B  EA9203                     JP PE,HIGH3
038E  3600              LOW3:    LD (HL),00H        ;value = 0
0390  1802                       JR NEXT3
0392  3610              HIGH3:   LD (HL),10H        ;value = 1
0394  2B                NEXT3:   DEC HL             ;move buffer pointer
0395  34                         INC (HL)           ;increment ADDL
0396  34                         INC (HL)           ;increment ADDL
0397  ED73E20F                   LD (DATAL),SP      ;copy SP to buffer
039B  21B90F                     LD HL,LEDL         ;set for CONVDI
039E  11E50F                     LD DE,ADDH         ;set for CONVDI
03A1  CD7CFA                     CALL CONVDI
03A4  CD09F9            DLOOP3:  CALL DISPL
03A7  DD3500                     DEC (IX+00)        ;timer for display
03AA  20F8                       JR NZ,DLOOP3
03AC  DD3502                     DEC (IX+02)        ;timer for display
03AF  20F3                       JR NZ,DLOOP3
03B1  DD3501                     DEC (IX+01)        ;timer for service routine
03B4  20CC                       JR NZ,CLOOP3
03B6  FDE1                       POP IY             ;restore CPU registers
03B8  DDE1                       POP IX
03BA  F1                         POP AF
03BB  E1                         POP HL
03BC  D1                         POP DE
03BD  C1                         POP BC
03BE  FB                         EI                 ;enable interrupts
03BF  ED4D                       RETI               ;return from interrupt
                        ;
                        ;
                        ;
                                 NAME INITOC
03C1  ED5E              INITOC:  IM2                ;set Z80 interrupt mode
03C3  21000F                     LD HL,TABLE        ;address of vector table
03C6  7C                         LD A,H             ;high byte of address
03C7  ED47                       LD I,A             ;set interrupt register
03C9  FD21E803                   LD IY,SERVOC       ;PIO output service routine
03CD  FD22060F                   LD (TABLE+06H),IY  ;set in vector table
03D1  3E06                       LD A,06H           ;Load interrupt vector
03D3  D30A                       OUT (0AH),A        ;for port C
03D5  08                         EX AF,AF'          ;set format for CONVDI
03D6  3E40                       LD A,40H
03D8  08                         EX AF,AF'
03D9  3E0F                       LD A,0FH           ;Set PIO mode
03DB  D30A                       OUT (0AH),A
03DD  3E87              ENPIO:   LD A,87H           ;Enable PIO interrupts
03DF  D30A                       OUT (0AH),A
03E1  3EFF                       LD A,0FFH          ;initialize CRDY signal
03E3  D308              THROW:   OUT (08H),A
03E5  C3C302                     JP MAIN            ;jump to routine MAIN
                        ;
                        ;
                        ;
                                 NAME SERVOC
```

```
03E8  E5         SERVOC:  PUSH HL           ;save CPU register status
03E9  F5                  PUSH AF
03EA  3AE40F              LD A,(ADDL)       ;move buffer value to A
03ED  D308                OUT (08H),A       ;output buffer value
03EF  F1                  POP AF            ;restore CPU register status
03F0  E1                  POP HL
03F1  FB                  EI                ;enable interrupts
03F2  ED4D                RETI              ;return from interrupt
                 ;
                 ;
                 ;
                          NAME INITID
03F4  ED5E       INITID:  IM2               ;Interrupt mode 2
03F6  21000F              LD HL,TABLE       ;address of vector table
03F9  7C                  LD A,H            ;high byte of address
03FA  ED47                LD I,A            ;set interrupt register
03FC  FD211F04            LD IY,SERVID      ;input service routine
0400  FD22080F            LD (TABLE+08H),IY ;set in vector table
0404  3E08                LD A,08H          ;Load interrupt vector
0406  D30B                OUT (0BH),A
0408  08                  EX AF,AF'         ;set format for CONVDI
0409  3E40                LD A,40H
040B  08                  EX AF,AF'
040C  3E4F                LD A,4FH          ;Set PIO mode
040E  D30B                OUT (0BH),A
0410  3E87                LD A,87H          ;enable PIO interrupt
0412  D30B                OUT (0BH),A
0414  DB09                   IN A,(09H)     ;initialize DRDY
0416  C3C302                 JP MAIN
                 ;
                 ;
                 ;
                          NAME SERVIC
0419  C5         SERVIC:  PUSH BC           ;save BC
041A  0E08                LD C,08H          ;PORT C interrupt
041C  C33104              JP SERVI
                 ;
                 ;
                 ;
                          NAME SERVID
041F  C5         SERVID:  PUSH BC
0420  0E09                LD C,09H          ;PORT D interrupt
0422  C33104              JP SERVI
                 ;
                 ;
                 ;
                          NAME SERVIE
0425  C5         SERVIE:  PUSH BC
0426  0E0C                LD C,0CH          ;PORT E interrupt
0428  C33104              JP SERVI
                 ;
                 ;
                 ;
                          NAME SERVIF
042B  C5         SERVIF:  PUSH BC
```

```
042C  0E0D                  LD C,0DH              ;PORT F Interrupt
042E  C33104                JP SERVI
                 ;
                 ;
                 ;
                            NAME SERVI
0431  00        SERVI:      NOP                   ;previously saved BC
0432  D5                    PUSH DE
0433  E5                    PUSH HL
0434  F5                    PUSH AF
0435  DDE5                  PUSH IX
0437  FDE5                  PUSH IY
0439  FD2AE40F              LD IY,(ADDL)          ;save state of (ADDL)
043D  FDE5                  PUSH IY
043F  ED78                  IN A,(C)
0441  32E40F                LD (ADDL),A           ;put byte in ADDL
0444  DD23      DSG:        INC IX                ;update data stack pointer
0446  DD23                  INC IX
0448  DD23                  INC IX
044A  00        ENABG:      NOP                   ;no operation
044B  DD3600FF              LD (IX+00H),0FFH      ;set DLOOPG time
044F  DD36010A              LD (IX+01H),00AH      ;set CLOOPG time
0453  DD360202  CLOOPG:     LD (IX+02H),02H       ;set DLOOPG time
0457  21E50F                LD HL,ADDH            ;point to display buffer
045A  ED57                  LD A,I                ;find value of IFF2
045C  EA6304                JP PE,HIGHG
045F  3600      LOWG:       LD (HL),00H           ;value = 0
0461  1802                  JR NEXTG
0463  3610      HIGHG:      LD (HL),10H           ;value = 1
0465  ED73E20F  NEXTG:      LD (DATAL),SP         ;copy SP to buffer
0469  21B90F                LD HL,LEDL            ;set for CONVDI
046C  11E50F                LD DE,ADDH            ;set for CONVDI
046F  CD7CFA                CALL CONVDI
0472  CD09F9    DLOOPG:     CALL DISPL
0475  DD3500                DEC (IX+00)           ;timer for display
0478  20F8                  JR NZ,DLOOPG
047A  DD3502                DEC (IX+02)           ;timer for display
047D  20F3                  JR NZ,DLOOPG
047F  DD3501                DEC (IX+01)           ;timer for service routine
0482  20CF                  JR NZ,CLOOPG
0484  FDE1                  POP IY                ;restore contents of ADDL
0486  FD22E40F              LD (ADDL),IY
048A  FDE1                  POP IY                ;restore CPU registers
048C  DDE1                  POP IX
048E  F1                    POP AF
048F  E1                    POP HL
0490  D1                    POP DE
0491  C1                    POP BC
0492  FB                    EI                    ;enable interrupts
0493  ED4D                  RETI                  ;return from interrupts
                 ;
                 ;
                 ;
                            NAME INITPB
0495  ED5E      INITPB:     IM2                   ;Z80 interrupt mode 2
```

```
0497  21000F              LD HL,TABLE          ;address of vector table
049A  7C                  LD A,H               ;high byte of address
049B  ED47                LD I,A               ;set interrupt register
049D  FD21E803            LD IY,SERVOC         ;output service routine
04A1  FF22060F            LD (TABLE+06H),IY    ;set in vector table
04A5  FD211904            LD IY,SERVIC         ;input service routine
04A9  FD220A0F            LD (TABLE+0AH),IY    ;set in vector table
04AD  3E06                LD A,06H             ;load interrupt vector
04AF  D30A                OUT (0AH),A          ;for port C
04B1  3E0A                LD A,0AH             ;load interrupt vector
04B3  D30B                OUT (0BH),A          ;for port D
04B5  08                  EX AF,AF'            ;set format for CONVDI
04B6  3E40                LD A,40H             ;
04B8  08                  EX AF,AF'            ;
04B9  3E8F                LD A,8FH             ;set PIO mode 2
04BB  D30A                OUT (0AH),A          ;port C
04BD  3ECF                LD A,0CFH            ;set PIO mode 3
04BF  D30B                OUT (0BH),A          ;port D
04C1  3EFF                LD A,0FFH            ;set mask byte Port D re-
04C3  D30B                OUT (0BH),A          ;quired to follow set PIO
04C5  3E87                LD A,87H             ;mode 3
04C7  D30A                OUT (0AH),A          ;enable PIO interrupts
                                               ;port C
04C9  D30B                OUT (0BH),A          ;port D
04CB  3EFF                LD A,0FFH            ;initialize CRDY
04CD  D308                OUT (08H),A
04CF  DB08                IN A,(08H)           ;initialize DRDY
04D1  C3C302              JP MAIN              ;jump to routine MAIN
                 ;
                 ;
                 ;
                          NAME INITPM
04D4  ED5E       INITPM:  IM2                  ;Z80 interrupt mode 2
04D6  21000F              LD HL,TABLE          ;address of vector table
04D9  7C                  LD A,H               ;high byte of address
04DA  ED47                LD I,A               ;set interrupt register
04DC  FD210505            LD IY,SERVM          ;address of service routine
04E0  FD220C0F            LD (TABLE+0CH),IY    ;set in vector table
04E4  3E0C                LD A,0CH             ;set interrupt vector for
04E6  D30B                OUT (0BH),A          ;port D
04E8  08                  EX AF,AF'            ;set format for CONVDI
04E9  3E40                LD A,40H
04EB  08                  EX AF,AF'
04EC  3ECF                LD A,0CFH            ;set mode 3 for port D
04EE  D30B                OUT (0BH),A
04F0  3E0F                LD A,0FH             ;define input lines for
04F2  D30B                OUT (0BH),A          ;port D
04F4  3E97       CWORD:   LD A,97H             ;set interrupt control word
04F6  D30B                OUT (0BH),A
04F8  3EFC                LD A,0FCH            ;monitor PB0,PB1
04FA  D30B                OUT (0BH),A
04FC  0E09                LD C,(09H)           ;initialize lamp monitors
04FE  3E00                LD A,00H             ;to off position
0500  ED79                OUT (C),A
0502  C3C302              JP MAIN
```

```
                  ;
                  ;
                  ;
                            NAME SERVM
0505  C5          SERVM:    PUSH BC               ;save CPU registers
0506  D5                    PUSH DE
0507  E5                    PUSH HL
0508  F5                    PUSH AF
0509  DDE5                  PUSH IX
050B  FDE5                  PUSH IY
050D  FD2AE40F              LD IY,(ADDL)          ;save state of (ADDL)
0511  FDE5                  PUSH IY
0513  0E09                  LD C,09H              ;input from PIO port C
0515  ED78                  IN A,(C)
0517  E60F                  AND 0FH               ;clear high order nibble
0519  32E40F                LD (ADDL),A           ;put byte in ADDL
051C  17                    RLA                   ;transpose high order
051D  17                    RLA                   ;nibble with low order
051E  17                    RLA                   ;nibble
051F  17                    RLA
0520  ED79                  OUT (C),A             ;output to lamp monitors
0522  DD23        DSM:      INC IX                ;update data stack pointer
0524  DD23                  INC IX
0526  DD23                  INC IX
0528  00                    NOP                   ;no operation
0529  DD3600FF              LD (IX+00H),0FFH      ;set inner DLOOPM time
052D  DD36010A              LD (IX+01H),00AH      ;set CLOOPM time
0531  DD360202    CLOOPM:   LD (IX+02H),02H       ;set outer DLOOPM time
0535  21E50F                LD HL,ADDH            ;point to display buffer
0538  ED57                  LD A,I                ;find value of IFF2
053A  EA4105                JP PE,HIGHM
053D  3600        LOWM:     LD (HL),00H           ;value = 0
053F  1802                  JR NEXTM
0541  3610        HIGHM:    LD (HL),10H           ;value = 1
0543  ED73E20F    NEXTM:    LD (DATAL),SP         ;copy SP to buffer
0547  21B90F                LD HL,LEDL            ;set for CONVDI
054A  11E50F                LD DE,ADDH            ;set for CONVDI
054D  CD7CFA                CALL CONVDI
0550  CD09F9      DLOOPM:   CALL DISPL
0553  DD3500                DEC (IX+00)           ;timer for display
0556  20F8                  JR NZ,DLOOPM
0558  DD3502                DEC (IX+02)           ;timer for display
055B  20F3                  JR NZ,DLOOPM
055D  DD3501                DEC (IX+01)           ;timer for service routine
0560  20CF                  JR NZ,CLOOPM
0562  FDE1                  POP IY                ;restore contents of ADDL
0564  FD22E40F              LD (ADDL),IY
0568  FDE1                  POP IY                ;restore CPU registers
056A  DDE1                  POP IX
056C  F1                    POP AF
056D  E1                    POP HL
056E  D1                    POP DE
056F  C1                    POP BC
0570  FB                    EI                    ;enable interrupts
0571  ED4D                  RETI                  ;return from interrupt
```

```
                    ;
                    ;
                    ;
                               NAME INITPP
0573    ED5E        INITPP:    IM2                    ;Z80 mode 2 interrupts
0575    21000F                 LD HL,TABLE            ;address of vector table
0578    7C                     LD A,H                 ;high byte of address
0579    ED47                   LD I,A                 ;set Interrupt vector
057B    FD211904               LD IY,SERVIC           ;service for port C
057F    FD220A0F               LD (TABLE+0AH),IY      ;set in table
0583    FD211F04               LD IY,SERVID           ;port D
0587    FD22080F               LD (TABLE+08H),IY      ;set in table
058B    3E0A                   LD A,0AH               ;set interrupt vector for C
058D    D30A                   OUT (0AH),A
058F    3E08                   LD A,08H               ;set interrupt vector for D
0591    D30B                   OUT (0BH),A
0593    08                     EX AF,AF'              ;set format for CONVDI
0594    3E40                   LD A,40H
0596    08                     EX AF,AF'
0597    3E4F                   LD A,4FH               ;mode 1 for C and D
0599    D30A                   OUT (0AH),A
059B    D30B                   OUT (0BH),A
059D    3E87                   LD A,87H               ;enable C and D
059F    D30A                   OUT (0AH),A
05A1    D30B                   OUT (0BH),A
05A3    DB08                   IN A,(08H)             ;initialize CRDY
05A5    DB09                   IN A,(09H)             ;and DRDY
05A7    C3C302                 JP MAIN
                    ;
                    ;
                    ;
                               NAME INITDC
05AA    ED5E        INITDC:    IM2                    ;Z80 interrupt mode 2
05AC    21000F                 LD HL,TABLE            ;address of vector table
05AF    7C                     LD A,H                 ;high byte of address
05B0    ED47                   LD I,A                 ;set Interrupt vector
05B2    FD212504               LD IY,SERVIE           ;service routine port E input
05B6    FD220E0F               LD (TABLE+0EH),IY      ;set in table
05BA    FD212B04               LD IY,SERVIF           ;service routine port F input
05BE    FD22100F               LD (TABLE+10H),IY      ;set in table
05C2    3E0E                   LD A,0EH               ;load interrupt vector E
05C4    D30E                   OUT (0EH),A
05C6    3E10                   LD A,10H               ;load interrupt vector F
05C8    D30F                   OUT (0FH),A
05CA    08                     EX AF,AF'              ;set format for CONVDI
05CB    3E40                   LD A,40H
05CD    08                     EX AF,AF'
05CE    3E4F                   LD A,4FH               ;set PIO mode 1
05D0    D30E                   OUT (0EH),A            ;port E
05D2    D30F                   OUT (0FH),A            ;port F
05D4    3E87                   LD A,87H               ;enable PIO
05D6    D30E                   OUT (0EH),A            ;port E
05D8    D30F                   OUT (0FH),A            ;port F
05DA    DB0C                   IN A, (0CH)            ;initialize ERDY
05DC    DB0D                   IN A, (0DH)            ;initialize FRDY
```

```
05DE  C37305                      JP INITPP
                        ;
                        ;
                        ;
                                  NAME SEROCX
05E1  C5              SEROCX:     PUSH BC           ;save CPU registers
05E2  D5                          PUSH DE
05E3  E5                          PUSH HL
05E4  F5                          PUSH AF
05E5  DDE5                        PUSH IX
05E7  FDE5                        PUSH IY
05E9  FD2AE40F                    LD IY,(ADDL)      ;save state of (ADDL)
05ED  FDE5                        PUSH IY
05EF  DD23            DSX:        INC IX            ;update data stack pointer
05F1  DD23                        INC IX
05F3  DD23                        INC IX
05F5  00                          NOP               ;no operation
05F6  DD3600FF                    LD (IX+00H),0FFH  ;set DLOOPX time
05FA  DD36010A                    LD (IX+01H),00AH  ;set CLOOPX time
05FE  DD360201        CLOOPX:     LD (IX+02H),01H   ;set DLOOPX time
0602  21E50F                      LD HL,ADDH        ;point to display buffer
0605  ED57                        LD A,I            ;find value of IFF2
0607  EA0E06                      JP PE,HIGHX
060A  3600            LOWX:       LD (HL),00H       ;value = 0
060C  1802                        JR NEXTX
060E  3610            HIGHX:      LD (HL),10H       value = 1
0610  2B              NEXTX:      DEC HL            ;move buffer pointer
0611  34                          INC (HL)          ;increment ADDL
0612  ED73E20F                    LD (DATAL),SP     ;copy SP to buffer
0616  21B90F                      LD HL,LEDL        ;set for CONVDI
0619  11E50F                      LD DE,ADDH        ;set for CONVDI
061C  CD7CFA                      CALL CONVDI
061F  CD09F9          DLOOPX:     CALL DISPL
0622  DD3500                      DEC (IX+00)       ;timer for display
0625  20F8                        JR NZ,DLOOPX
0627  DD3502                      DEC (IX+02)       ;timer for display
062A  20F3                        JR NZ,DLOOPX
062C  DD3501                      DEC (IX+01)       ;timer for service routine
062F  20CD                        JR NZ,CLOOPX
0631  FDE1                        POP IY            ;restore CPU registers
0633  FD22E40F                    LD (ADDL),IY      ;restore state of (ADDL)
0637  3AE40F          OUTX:       LD A,(ADDL)       ;output the byte that was
063A  D308                        OUT (08H),A       ;in ADDL when interrupted
063C  FDE1                        POP IY            ;restore CPU registers
063E  DDE1                        POP IX
0640  F1                          POP AF
0641  E1                          POP HL
0642  D1                          POP DE
0643  C1                          POP BC
0644  FB                          EI                ;enable interrupts
0645  ED4D                        RETI              ;return from interrupt
                        ;
                        ;
                        ;
                                  NAME CHPTST
```

```
0647  3E03                CHPTST:  LD A,03H          ;Reset PIO interrupt
                                                     ;enable FLIP-FLOP
0649  D30A                         OUT (0AH),A
064B  D30B                         OUT (0BH),A
064D  2A0300                       LD HL,(MASKW)     ;SET mask for circuit
0650  010AFF                       LD BC,0FF0AH
0653  ED41                         OUT (C),B         ;PIO Port A to mode 3
0655  ED69                         OUT (C),L         ;I/O mask for Port A
0657  0C                           INC C             ;Change to Port B
0658  ED41                         OUT (C),B         ;PIO Port B to mode 3
065A  ED69                         OUT (C),H         ;I/O mask for Port B
                                                     ;
                                                     ;
                                                     ;
                                                     ;
                                                     ;
                                                     ;
                          ;
065C  31A00F              REF:     LD SP,CHPSTK      ;Initialize
                                                     ;stack pointer
065F  DD210008                     LD IX,REFIC       ;Initialize
                                                     ;reference IC
                                                     ;map pointer
0663  010000                       LD BC,0000H       ;Initialize
                                                     ;counter word
0666  CD8806                       CALL STORE        ;Generate the
                                                     ;reference table
0669  00                  ENDREF:  NOP
                          ;
066A  31A00F              UNKN:    LD SP,CHPSTK      ;Initialize
                                                     ;stack pointer
066D  DD21000C                     LD IX,UNKIC       ;Initialize un-
                                                     ;known IC map
                                                     ;pointer
0671  010000                       LD BC,0000H       ;Initialize
                                                     ;counter word
0674  CD8806                       CALL STORE        ;Generate the
                                                     ;unknown IC's
                                                     ;output table
                          ;
0677  210008              COMPAR:  LD HL,REFIC       ;Set-up for com-
                                                     ;pare using the
                                                     ;CPI instruction
067A  11000C                       LD DE,UNKIC       ;HL points to
                                                     ;ref table, DE
                                                     ;points to unk
                                                     ;IC table
067D  1A                  NEXTB:   LD A,(DE)         ;Load unknown
                                                     ;output byte into
                                                     ;accumulator
067E  EDA1                         CPI               ;Compare with (HL)
0680  2037                         JR NZ,BAD         ;If not =, we have
                                                     ;a bad IC
0682  13                           INC DE            ;If =, set up to.
                                                     ;test next bye
```

```
0683  EA7D06              JP PE,NEXTB      ;If P/V flag = 1
                                           ;go test next byte
0686  1833     GOOD:      JR START         ;If P/V flag = 0
                                           ;BC is zero and
                                           ;we have tested
                                           ;all the bytes
               ;
0688  110000   STORE:     LD DE,0000H      ;Initialize test
                                           ;word
068B  2A0300   NTEST:     LD HL,(MASKW)    ;Load HL with mask
                                           ;word
068E  7B                  LD A,E           ;Perform 16-bit
                                           ;AND on mask and
                                           ;test words
068F  A5                  AND L
0690  6F                  LD L,A
0691  7A                  LD A,D
0692  A4                  AND H
0693  67                  LD H,A
0694  7C       MASK:      LD A,H           ;Check if result
                                           ;of 16-bit AND = 0
0695  B5                  OR L
0696  201B                JR NZ,NXTWD      ;If not 0, go to
                                           ;next test byte
               ;
0698  7B       TEST:      LD A,E           ;If = 0, it is a
                                           ;valid test word.
                                           ;Output it to IC
0699  D308                OUT (08H),A
069B  7A                  LD A,D
069C  D309                OUT (09H),A
069E  2A0300              LD HL,(MASKW)    ;Get mask word for
                                           ;IC
06A1  DB08                IN A,(08H)       ;Input LO byte
                                           ;from IC
06A3  A5                  AND L            ;Mask it
06A4  DD7700              LD (IX),A        ;Store it
06A7  DD23                INC IX           ;Update IX
06A9  DB09                IN A,(09H)       ;Input HI byte
                                           ;from IC
06AB  A4                  AND H            ;Mask it
06AC  DD7700              LD (IX),A        ;Store it
06AF  DD23                INC IX           ;Update IX
06B1  03                  INC BC           ;Add two to
                                           ;counter
06B2  03                  INC BC
               ;
06B3  13       NXTWD:     INC DE           ;Get next test
                                           ;word
06B4  7A                  LD A,D
06B5  B3                  OR E
06B6  20D3                JR NZ,NTEST      ;If DE is not
                                           ;zero, go back
                                           ;for next test
                                           ;word
```

```
06B8   C9                      RET               ;If DE is zero
                                                 ;full output
                                                 ;table is gen-
                                                 ;erated
06B9   1800         BAD:       JR START          ;Bad IC, start
                                                 ;over
                    ;
06BB   18AD         START:     JR UNKN           ;Jump to test
                                                 ;routine for
                                                 ;unknown IC
                    ;
                    ;
                    ;
                               NAME INITC1
06BD   ED5E         INITC1:    IM2               ;Z80 Interrupt Mode 2
06BF   21000F                  LD HL,TABLE       ;address of vector table
06C2   7C                      LD A,H            ;high byte of address
06C3   ED47                    LD I,A            ;set interrupt register
06C5   FD216E02                LD IY,SERV1       ;service routine address
06C9   FD221A0F                LD (TABLE+1AH),IY ;set in table
06CD   3E18                    LD A,18H          ;load interrupt vector
06CF   D310                    OUT (10H),A       ;to CTC CHANNEL 0
06D1   08                      EX AF,AF'         ;set format for CONVDI
06D2   3E40                    LD A,40H
06D4   08                      EX AF,AF'
06D5   3EC7                    LD A,0C7H         ;set channel control word
06D7   D311                    OUT (11H),A
06D9   3E05                    LD A,05H          ;set time constant
06DB   D311                    OUT (11H),A       ;register
06DD   C3C302                  JP MAIN           ;jump to routine MAIN
                    ;
                    ;
                    ;
                               NAME SERCT1
06E0   C5           SERCT1:    PUSH BC           ;save status of BC
06E1   0E11                    LD C,11H          ;PORT 11H of CTC
06E3   C33104                  JP SERVI
                    ;
                    ;
                    ;
                               NAME SERCT2
06E6   C5           SERCT2:    PUSH BC           ;save CPU registers
06E7   D5                      PUSH DE
06E8   E5                      PUSH HL
06E9   F5                      PUSH AF
06EA   DDE5                    PUSH IX
06EC   FDE5                    PUSH IY
06EE   FD2AE40F                LD IY,(ADDL)      ;save state of (ADDL)
06F2   FDE5                    PUSH IY
06F4   0E16                    LD C,16H          ;input from CTC
06F6   ED40                    IN B,(C)
06F8   AF                      XOR A             ;clear A
06F9   90                      SUB B             ;find number of seconds
06FA   32E40F                  LD (ADDL),A       ;load ADDL with CTC data
06FD   DD23         DST:       INC IX            ;update data stack pointer
```

```
06FF  DD23              INC IX
0701  DD23              INC IX
0703  00                NOP                     ;no operation
0704  DD3600FF          LD (IX+00H),0FFH        ;set DLOOPT time
0708  DD36010A          LD (IX+01H),00AH        ;set CLOOPT time
070C  DD360202  CLOOPT: LD (IX+02H),02H         ;set DLOOPT time
0710  21E50F            LD HL,ADDH              ;point to display buffer
0713  ED57              LD A,I                  ;find value of IFF2
0715  EA1C07            JP PE,HIGHT
0718  3600      LOWT:   LD (HL),00H             ;value = 0
071A  1802              JR NEXTT
071C  3610      HIGHT:  LD (HL),10H             ;value = 1
071E  ED73E20F  NEXTT:  LD (DATAL),SP           ;copy SP to buffer
0722  21B90F            LD HL,LEDL              ;set for CONVDI
0725  11E50F            LD DE,ADDH              ;set for CONVDI
0728  CD7CFA            CALL CONVDI
072B  CD09F9    DLOOPT: CALL DISPL
072E  DD3500            DEC (IX+00)             ;timer for display
0731  20F8              JR NZ,DLOOPT
0733  DD3502            DEC (IX+02)             ;timer for display
0736  20F3              JR NZ,DLOOPT
0738  DD3501            DEC (IX+01)             ;timer for service routine
073B  20CF              JR NZ,CLOOPT
073D  3E2F              LD A,2FH                ;Channel 0 control word
073F  D314              OUT (14H),A
0741  3E96              LD A,96H                ;Channel 0 time constant
0743  D314              OUT (14H),A
0745  3E47              LD A,47H                ;Channel 1 control word
0747  D315              OUT (15H),A
0749  3E40              LD A,40H                ;Channel 1 time constant
074B  D315              OUT (15H),A
074D  3E47              LD A,47H                ;Channel 2 control word
074F  D316              OUT (16H),A
0751  3E00              LD A,00H                ;Channel 2 time constant
0753  D316              OUT (16H),A
0755  3EC7              LD A,0C7H               ;Channel 3 control word
0757  D317              OUT (17H),A
0759  3E01              LD A,01H                ;Channel 3 time constant
075B  D317              OUT (17H),A
075D  FDE1              POP IY                  ;restore contents of ADDL
075F  FD22E40F          LD (ADDL),IY
0763  FDE1              POP IY                  ;restore CPU registers
0765  DDE1              POP IX
0767  F1                POP AF
0768  E1                POP HL
0769  D1                POP DE
076A  C1                POP BC
076B  FB                EI                      ;enable interrupt flip-flop
076C  ED4D              RETI                    ;return from interrupts
                ;
                ;
                ;
                        NAME INITC3
076E  ED5E      INITC3: IM2                     ;Z80 Interrupt Mode 2
0770  21000F            LD HL,TABLE             ;vector address table
```

```
0773   7C                    LD A,H             ;high byte of address
0774   ED47                  LD I,A             ;set interrupt register
0776   FD21E606              LD IY,SERCT2       ;service routine address
077A   FD22260F              LD (TABLE+26H),IY  ;set in table
077E   3E26                  LD A,26H           ;load interrupt vector
0780   D314                  OUT (14H),A        ;to CTC Channel 0
0782   08                    EX AF,AF'          ;set format for CONVDI
0783   3E40                  LD A,40H
0785   08                    EX AF,AF'
0786   3E2F                  LD A,2FH           ;Channel 0 control word
0788   D314                  OUT (14H),A
078A   3E96                  LD A,96H           ;Channel 0 time constant
078C   D314                  OUT (14H),A
078E   3E47                  LD A,47H           ;Channel 1 control word
0790   D315                  OUT (15H),A
0792   3E40                  LD A,40H           ;Channel 1 time constant
0794   D315                  OUT (15H),A
0796   3E47                  LD A,47H           ;Channel 2 control word
0798   D316                  OUT (16H),A
079A   3E00                  LD A,00H           ;Channel 2 time constant
079C   D316                  OUT (16H),A
079E   3EC7                  LD A,0C7H          ;Channel 3 control word
07A0   D317                  OUT (17H),A
07A2   3E01                  LD A,01H           ;Channel 3 time constant
07A4   D317                  OUT (17H),A
07A6   C3C302                JP MAIN
                  ;
                  ;
                  ;
                             NAME INITC2
07A9   FD21E006   INITC2:    LD IY,SERCT1       ;service routine address
07AD   FD22180F              LD (TABLE+18H),IY  ;set in table
07B1   3EC7                  LD A,0C7H          ;channel 0 control word
07B3   D310                  OUT (10H),A
07B5   3E01                  LD A,01H           ;time constant register
07B7   D310                  OUT (10H),A        ;for channel 0
07B9   C3BD06                JP INITC1
                  ;
                  ;
                  ;
07BC   (0010)                DS 10H
                  ;
                  ;
                  ;
07CC                         ORG 0F000H
                             NAME BLKMVE
       (0100)     ORIGIN     EQU 100H
       (0700)     LENGHT     EQU 0700H
                  ;
                  ;
                  ;
F000   FB         BLKMVE:    EI
F001   218EF0                LD HL,RESTART
F004   F3                    DI
F005   110001                LD DE,ORIGIN
```

```
F008  010007               LD BC,LENGHT
F00B  EDB0                 LDIR
                ;
                ;
                ;
                ;
                ;
                ;
                           NAME NANOR2
F00D  DD21000C  NANOR2:    LD IX,DSTACK        ;Set IX to RAM
                                               ;counter location
F011  2142F0               LD HL,STRING
F014  11B80F    MOVE:      LD DE,LEDH          ;and DE to dis-
                                               ;play buffer
F017  010A00               LD BC,0AH           ;BC=no. of bytes
                                               ;to move
F01A  E5                   PUSH HL             ;Save character
                                               ;pointer
F01B  EDB0                 LDIR                ;Move first 10
                                               ;bytes
F01D  DD3600FF             LD (IX),0FFH        ;Preset counter
F021  DD360101             LD (IX+1H),01H      ;for display scan
                                               ;speed
F025  3E00                 LD A,00H
F027  32B80F               LD (LEDH),A         ;Mask off LED dis-
                                               ;plays
F02A  32B90F               LD (LEDH+1H),A
F02D  CD09F9    DS:        CALL DISPL
F030  DD3500               DEC (IX)            ;Time . . .
F033  20F8                 JR NZ,DS            ;  . . . delay
F035  DD3501               DEC (IX+1H)         ;. . . and
F038  20F3                 JR NZ,DS            ;      . . . display
F03A  E1                   POP HL              ;Retrieve char-
                                               ;acter pointer
                                               ;value
F03B  23                   INC HL              ;And increment
F03C  7E                   LD A,(HL)           ;Check charac-
                                               ;ter for end-
                                               ;code
F03D  FE01                 CP 01H              ;'01', otherwise
                                               ;move along
F03F  20D3                 JR NZ,MOVE
F041  FF                   RST 38H             ;Return control
                                               ;to the Nanocomputer
                                               ;operating system
                ;
                ;
                ;
F042  00000000  STRING:    DB 00H,00H,00H,00H        ;Leading blanks
F046  00000000             DB 00H,00H,00H,00H
F04A  0000B6BC             DB 00H,00H,0B6H,0BCH      ;SG
F04E  B602EE1E             DB 0B6H,02H,0EEH,1EH      ;S-AT
F052  9EB600EC             DB 9EH,0B6H,00H,0ECH      ;ES N
F056  EEECFC00             BD 0EEH,0ECH,0FCH,00H     ;AN0
F05A  0A3A381E             DB 0AH,3AH,38H,1EH        ;ROUT
```

```
F05E    202A9EB6             DB 20H,2AH,9EH,0B6H     ;INES
F062    000A9E1C             DB 00H,0AH,9EH,1CH      ;  REL
F066    9EEEB69E             DB 9EH,0EEH,0B6H,9EH    ;EASE
F06A    00DA02DA             D8 00H,0DAH,02H,0DAH    ;  2-2
F06E    001CFCEE             DB 00H,1CH,0FCH,0EEH    ;  LOA
F072    7A9E7A00             DB 7AH,9EH,7AH,00H      ;DED
F076    00000000             DB 00H,00H,00H,00H      ;
F07A    009C60EE             DB 00H,9CH,60H,0EEH     ;  CIA
F07B    FC000010             DB 0FCH,00H,00H,10H     ;O  -
F082    00100110             DB 00H,10H,01H,10H      ;  -  -
F086    00000000             DB 00H,00H,00H,00H      ;Trailing blanks
F08A    00000000             DB 00H,00H,00H,00H
                     RESTART:
                     ;
                     ;
                     ;
```

APPENDICE C

PRECAUZIONI DA ADOTTARE NEL MANEGGIARE DISPOSITIVI MOS

I dispositivi MOS sono estremamente sensibili e possono essere danneggiati da:

Elettricità statica

Inserzione errata negli zoccoli della scheda Nanocomputer.

Nel maneggiare dispositivi MOS, dovrebbero essere osservate le precauzioni seguenti:

1. Assicurarsi di essere elettricamente scarichi prima di toccare il dispositivo. Ciò può essere fatto strofinando le mani sul materiale conduttore.
2. Evitare di toccare i piedini del dispositivo.
3. Evitare di mettere a contatto i piedini con qualsiasi materiale capace di caricarsi elettrostaticamente (ad esempio, tappeti di fibra sintetica).
4. Se è necessario trasportare un dispositivo MOS mettere un supporto conduttore, per evitare che tra i diversi piedini si possano avere delle differenze di potenziale.
5. Quando si deve sostituire un dispositivo, assicurarsi di inserire il nuovo dispositivo in modo corretto, in particolare assicurarsi che il piedino 1 si trovi al posto giusto.

PRECAUZIONI DA ADOTTARE NEL MANEGGIARE DISPOSITIVI MOS

I dispositivi MOS sono [illegible] e possono essere danneggiati da:

- Elettricità statica;
- [illegible]

Nel maneggiare dispositivi MOS, devono essere osservate le precauzioni seguenti:

1. Assicurarsi di essere [illegible] prima di toccare il dispositivo. [illegible]
2. Evitare di toccare i piedini dei dispositivi.
3. Evitare di mettere a contatto i piedini [illegible]
4. Se è necessario trasportare dei dispositivi MOS [illegible]
5. Quando si [illegible] dispositivi, [illegible]

APPENDICE D

SCHEMI ELETTRICI DEL NANOCOMPUTER

Le pagine seguenti contengono un insieme completo degli schemi del Nanocomputer. Riferitevi allo schema appropriato, a seconda della sezione del Nanocomputer che state analizzando nell'ambito del testo. I primi due schemi si riferiscono alla scheda base contenente la CPU, i successivi, alla scheda per esperimenti e al mini-terminale tastiera/display.

CPU

Q 29
Z80 CPU

M1 27, MREQ 19, IORQ 20, RD 21, WR 22, RFSH 28, HALT 18, BWAIT 24, BINT 16, BNMI 17, RESET 26, BBUSRQ 25, BUSAK 23, ∅ 6

30 AD∅, 31 AD1, 32 AD2, 33 AD3, 34 AD4, 35 AD5, 36 AD6, 37 AD7, 38 AD8, 39 AD9, 40 AD1∅, 1 AD11, 2 AD12, 3 AD13, 4 AD14, 5 AD15

14 D∅, 15 D1, 12 D2, 8 D3, 7 D4, 9 D5, 10 D6, 13 D7

29, 11 +5V

BUSAK 13 74LS14 Q11 12

74LS365 Q40: AD∅ → BAD∅ (J1-2c), AD1 → BAD1 (J1-7c), AD2 → BAD2 (J1-6c), AD3 → BAD3 (J1-1c), AD4 → BAD4 (J1-4c), AD5 → BAD5 (J1-5c)

74LS365 Q41: AD6 → BAD6 (J2-30c), AD7 → BAD7 (J2-29c), AD8 → BAD8 (J2-31c), AD9 → BAD9 (J2-28c), AD1∅ → BAD1∅ (J1-12c), AD11 → BAD11 (J2-27c)

74LS367 Q42: AD12 → BAD12 (J2-19c), AD13 → BAD13 (J2-7c), AD14 → BAD14 (J2-18c), AD15 → BAD15 (J2-17c), D6 → BD6 (J2-12c), D7 → BD7 (J2-10c), DBOUT 15

EPROM PROM ROM MEMORY

Q 49 — 2708 / 6381 / 2716 / 2316 E
Q 50 — 2708 / 6381 / 2716 / 2316 E
Q 51 — 2708 / 6381 / 2716 / 2316 E
Q 52 — 2708 / 6381 / 2716 / 2316 E

BAD∅ 8, BAD1 7, BAD2 6, BAD3 5, BAD4 4, BAD5 3, BAD6 2, BAD7 1, BAD8 23, BAD9 22, CS∅/CS1/CS2/CS3 20, VDD-E3-A10 19

9 D∅, 10 D1, 11 D2, 13 D3, 14 D4, 15 D5, 16 D6, 17 D7, 24 +5V, 21 VBB-E1-VPP-CS3, 18 PRO-E4-PGM-CS2, 12

RAM CONTROL

Q37 74LS157: BAD11 2 I∅A, BAD4 3 I1A, 5 I∅B, BAD5 6 I1B, 14 I∅C, 13 I1C, 11 I∅D, ICAS 10 I1D; ZA 4 MA4, ZB 7 MA5, ZC 12 CS-MA6, ZD 9 CAS, S 1 SELAD, E 15

Q47 74LS157: BAD7 2 I∅A, BAD∅ 3 I1A, BAD8 5 I∅B, BAD1 6 I1B, BAD9 14 I∅C, BAD2 13 I1C, BAD1∅ 11 I∅D, BAD3 10 I1D; ZA 4 MA∅, ZB 7 MA1, ZC 12 MA2, ZD 9 MA3, S 1 SELAD, E 15

41, 44, 40, 43 BAD13, 39, 42 BAD12, BAD6

ROM CONTROL

BAD1∅ 9, BAD12 10

Q38 74LS139: BAD11 3 A∅A, 2 A1A, 1 EA, CS∅ 4 O∅A, CS1 5 O1A, CS2 6 O2A, CS3 7 O3A; 14 A∅B BAD∅, 13 A1B BAD1, 15 EB BIORQ, 12 O∅B IOU∅, 11 O1B IOU1 (J1-24c), 10 O2B IOU2 (J1-23c), 9 O3B IOU3 (J1-22c) (J1-21c), IOU3

PAGRO 2, BRFSH 13, 74LS∅∅ Q35 11, ROMSEL

DEVICE SELECT

Q53 74LS138: IOQ3 12 O3, IOQ2 13 O2, IOQ1 14 O1, IOQ∅ 15 O∅, BAD4 3 A2, BAD3 2 A1, BAD2 1 A∅; 11 O4 IOE∅, 10 O5 IOE1 (J2-6c), 9 O6 IOE2 (J2-8c), 7 O7 IOE3 (J2-9c), 6 E3, 5 E2, 4 E1 (J2-5c) (J2-11c)

BAD7 13 Q31 12, 32, 33, 74LS∅4 34, 35, 74LS∅4 36, BAD6 5 Q31 6, 37, 38, 66, BAD5

RAM MEMORY

Q27 4027, Q26 4027, Q25 4027, Q24 4027, Q23 4027, Q22 4027, Q21 4027, Q20 4027

MA∅ 5, MA1 7, MA2 6, MA3 12, MA4 11, MA5 10, CS-MA6 13, D∅/D1/D2/D3/D4/D5/D6/D7 2

4 RAS, 15 CAS, 3 WRM, 8 +12V, 16, 1 -5V, 9 +5V, 14 MO∅/MO1/MO2/MO3/MO4/MO5/MO6/MO7

74LS367 Q28: MO∅ 4 → 5 D∅, MO1 2 → 3 D1, MO2 10 → 9 D2, MO3 6 → 7 D3, LETRA 1

74LS367 Q19: MO4 10 → 9 D4, MO5 6 → 7 D5, MO6 4 → 5 D6, MO7 2 → 3 D7, LETRA 1

BUS INTERFACE

MEMORY SELECT

RAM TIMING

* : IC'S OMITTED ON THIS ASSEMBLY VERSION. (See part list for exact details)

* : COMPONENTI OMESSI IN QUESTA VERSIONE (Vedi lista componenti).

USED ONLY FOR TESTING

SCHEMATIC DIAGRAM
SCHEMA ELETTRICO

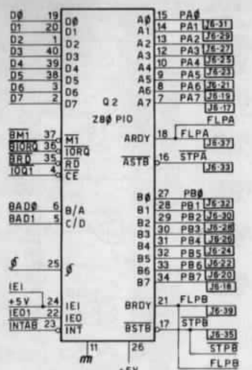
CLZ80-NC

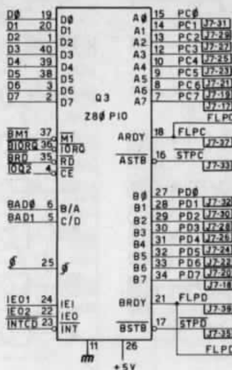
EEU003.26.V07

SHEET 1 OF 2

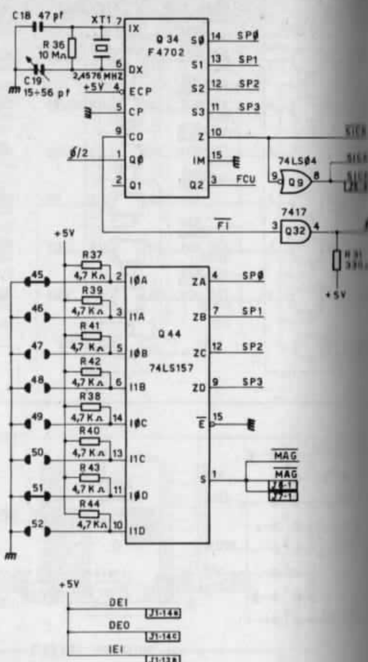
12-12-79

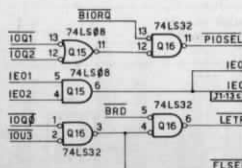
INPUT-OUTPUT PORTS A-B-C-D

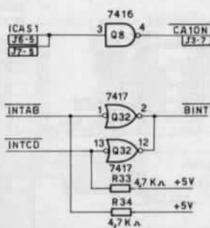




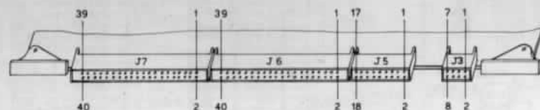
CLOCK AND BAUD GENERATOR

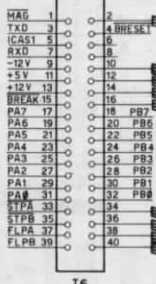


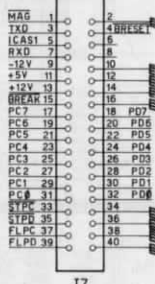




+5V
DEI
DEO
IEI







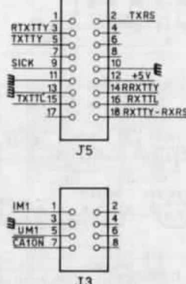


TABLE 1

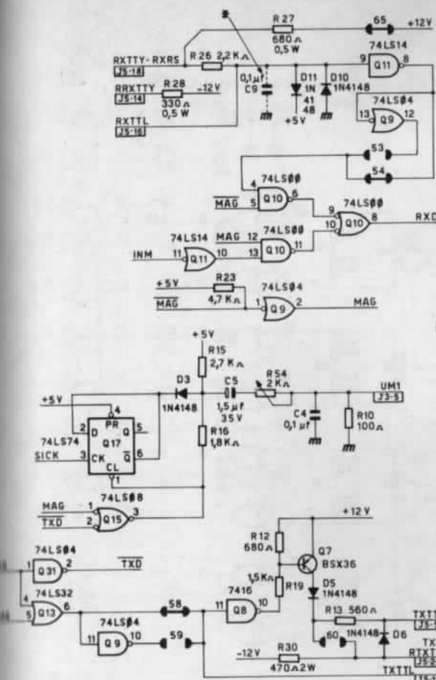
EPROM-PROM ROM SELECT	JUMPERS
2708	1.3.6
6381	2.4.7
2716 .1(2)	68.5.8
2716	1.5.8
2316 E	1.5.8

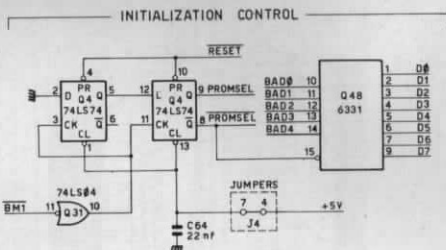
TABLE 2

4K EPROM PROM-ROM PARTITION START-END		JUMPERS			
		9 26	9 25	9 24	9 23
JUMPERS	30 15	ØK 4K	4K 8K	8K 12K	12K 16K
	30 16	16K 20K	20K 24K	24K 28K	28K 32K
	30 17	32K 36K	36K 40K	40K 44K	44K 48K
	30 18	48K 52K	52K 56K	56K 60K	60K 64K

TABLE 3

8K EPROM PROM-ROM PARTITION START-END		JUMPERS			
		15 10	16 10	17 10	18 10
JUMPERS	27 30	ØK 8K	16K 24K	32K 40K	48K 56K
	28 29	8K 16K	24K 32K	40K 48K	56K 64K





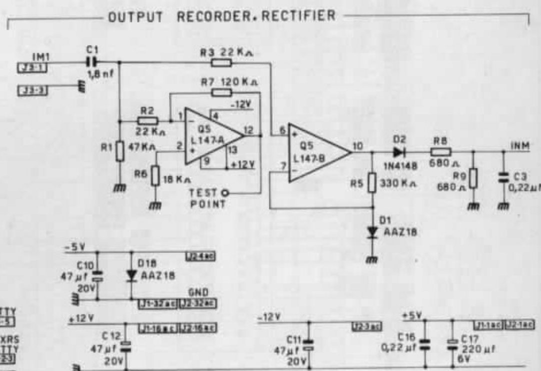


TABLE 4

RAM CONNECTIONS		JUMPERS	SIZE
DEVICE	4027	39.40.41	4K x 8
	4116	42.43.44	16K x 8

TABLE 5

4K-RAM PARTITION START-END		JUMPERS 22	21	20	19
JUMPERS	11	ØK / 4K	4K / 8K	8K / 12K	12K / 16K
	12	16K / 20K	20K / 24K	24K / 28K	28K / 32K
	13	32K / 36K	36K / 40K	40K / 44K	44K / 48K
	14	48K / 52K	52K / 56K	56K / 60K	60K / 64K

TABLE 6

16K-RAM PARTITION	JUMPERS 31 11	31 12	31 13	31 14
START-END	ØK / 16K	16K / 32K	32K / 48K	48K / 64K

TABLE 8

ASCII SERIAL INTERFACE	TRANSMISSION LINE TYPE TTY	RS232	TTL
JUMPERS	65 54 58	60 54 59	53 58
SERIAL INPUT	RXTTY	RXRS	RXTTL
RETURN	RRXTTY	GND	GND
SERIAL OUTPUT	TXTTY	TXRS	TXTTL
RETURN	RTXTTY	GND	GND

TABLE 7

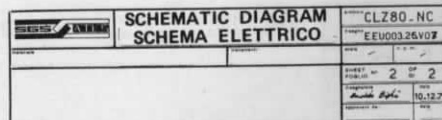
INITIAL DEVICE CODE SELECTOR		JUMPERS 36 37	34 38	36 66	35 38
JUMPERS	32	Ø	32	64	96
	33	128	160	192	224

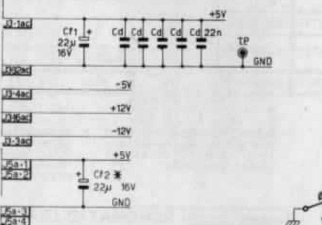
TABLE 9

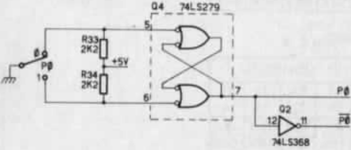
BAUD RATE		CASS.	
600		51	45

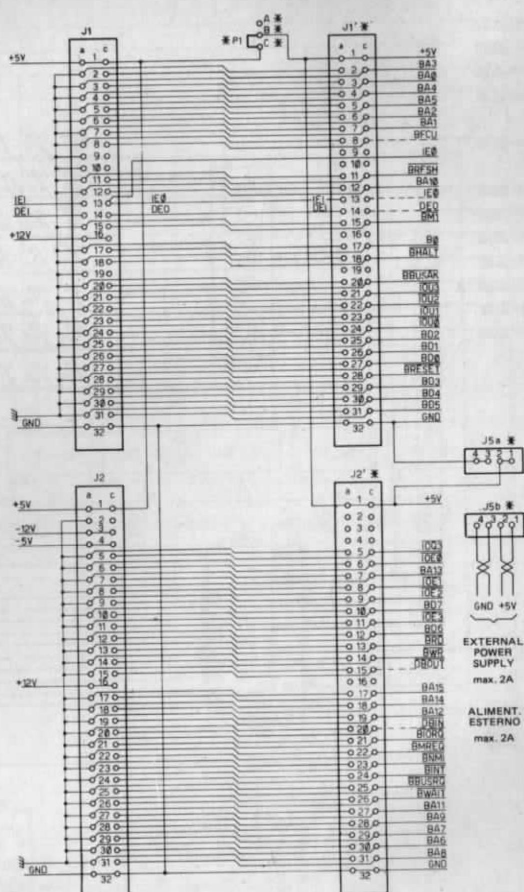
* : INSERT C9 CAPACITOR WHEN THE SERIAL LINE DEVICE IS A TELETYPEWRITER WITH MECHANICAL CONTACTS.

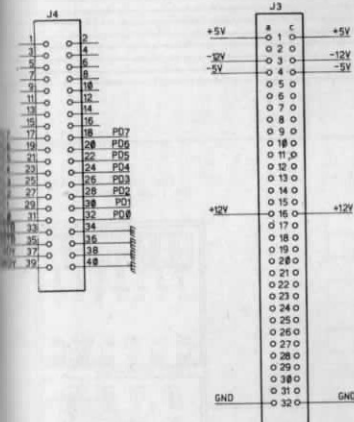
* : INSERIRE IL CONDENSATORE C9 QUANDO SULLA LINEA SERIALE VIENE CONNESSA UNA TELESCRIVENTE A CONTATTI MECCANICI.

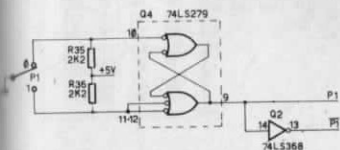












SIGNAL CONVERSION TABLE TAB. CONVERSIONE SEGNALI	
CLZ8Ø/NC	NEZ8Ø
BADØ ÷ BAD15	BAØ – BA15
FLPC	CRDY
STPC	CSTB
FLPD	DRDY
STPD	DSTB
IEO	IEØ

NOTE:
THE JUMPER P1 IS PLACED BETWEEN THE POINTS B-C ONLY WHEN THE BOARD CONNECTED TO THE gamma-BUS HAS AN CONSUMPTION ≤ 0,5A.
WHEN THE BOARD CONNECTED TO THE gamma-BUS CONSUMES MORE THAN 0,5A THE JUMPER P1 MUST BE PLACED BETWEEN THE POINTS B-A.
IN THIS CASE THE BOARD CONNECTED TO THE gamma-BUS IS SUPPLIED BY AN EXTERNAL POWER SUPPLY (max. 2A) THROUGH THE APPROPRIATE CONNECTOR (J5a).

* : COMPONENTS OMITTED ON THIS ASSEMBLY VERSION.

NOTA:
IL CAVALLOTTO P1 VA INSERITO TRA I PUNTI B-C SOLO QUANDO LA SCHEDA COLLEGATA AL gamma-BUS HA UN ASSORBIMENTO ≤ 0,5A.
QUANDO LA SCHEDA COLLEGATA AL gamma-BUS ASSORBE PIU' DI 0,5A IL CAVALLOTTO P1 DEVE ESSERE COLLEGATO TRA I PUNTI B-A.
IN QUESTO CASO LA SCHEDA COLLEGATA AL gamma-BUS E' ALIMENTATA DA UN ALIMENTATORE ESTERNO (MASSIMO 2A) PER MEZZO DEL RELATIVO CONNETTORE (J5a).

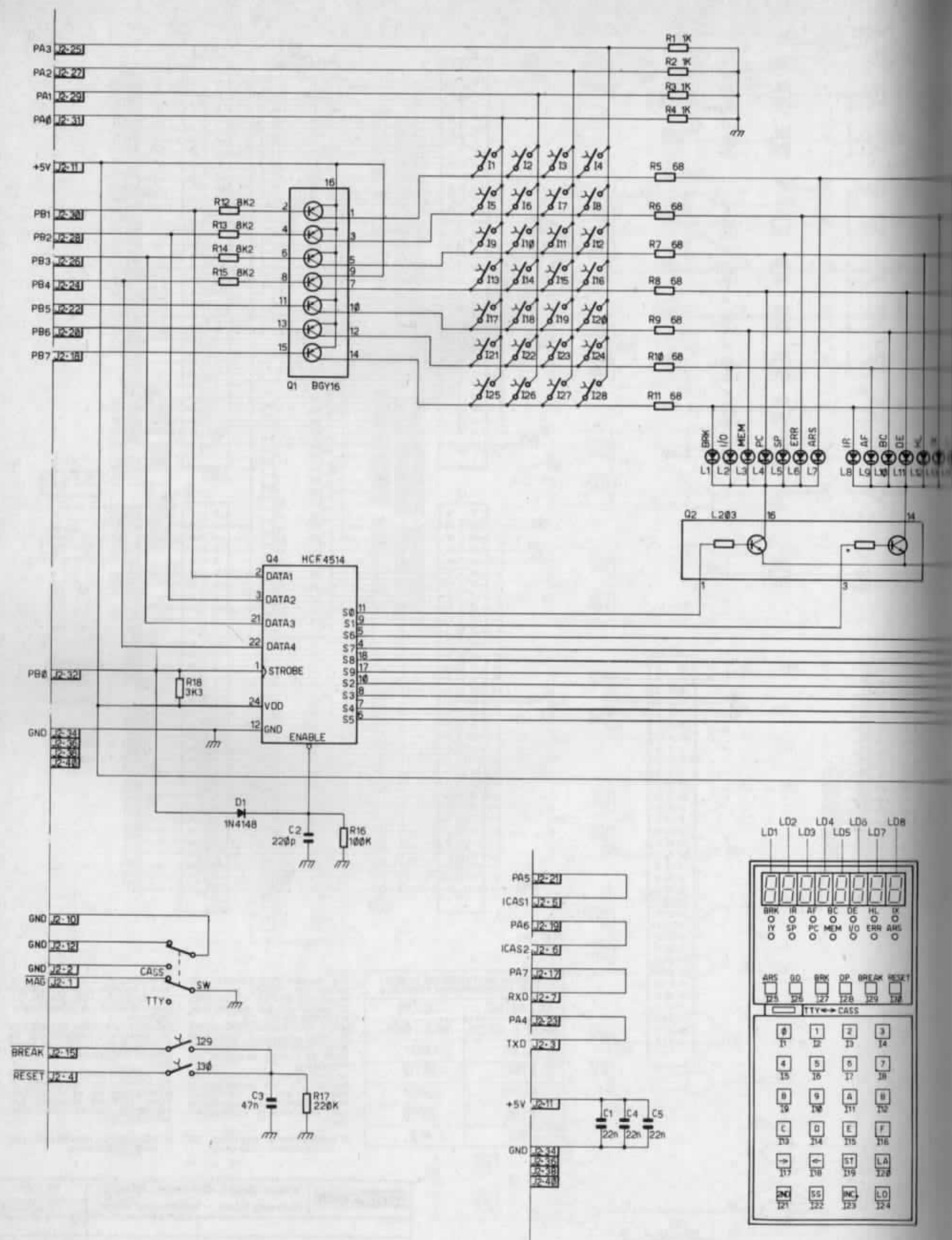
* : COMPONENTI NON MONTATI IN QUESTA VERSIONE.

SGS-ATES

Exper. Board – Schematic Diagram
Schema Elett. – Scheda per esper.

NEZ80
EEU01501V03

1 / 1
5-2-80

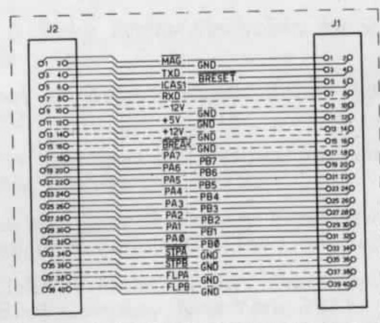


PA3 J2-25
PA2 J2-27
PA1 J2-29
PAØ J2-31
+5V J2-11
PB1 J2-30
PB2 J2-28
PB3 J2-26
PB4 J2-24
PB5 J2-22
PB6 J2-20
PB7 J2-18
R1 1K
R2 1K
R3 1K
R4 1K
R5 68
R6 68
R7 68
R8 68
R9 68
R1Ø 68
R11 68
R12 8K2
R13 8K2
R14 8K2
R15 8K2
Q1 BGY16
Q2 L2Ø3
Q4 HCF4514
DATA1
DATA2
DATA3
DATA4
STROBE
VDD
GND
ENABLE
PBØ J2-32
R18 3K3
GND J2-34
D1 1N4148
C2 22Øp
R16 1ØØK
BRK I/O MEM PC SP ERR ARS
IR AF BC DE
LD1 LD2 LD3 LD4 LD5 LD6 LD7 LD8
GND J2-10
GND J2-12
GND J2-2
MAG J2-1
CASS
SW
TTY
BREAK J2-15
RESET J2-4
I29
I3Ø
C3 47n
R17 22ØK
PA5 J2-21
ICAS1 J2-5
PA6 J2-19
ICAS2 J2-6
PA7 J2-17
RXD J2-7
PA4 J2-23
TXD J2-3
+5V J2-11
C1 22n
C4 22n
C5 22n
GND J2-34
ARS GO BRK OP BREAK RESET
TTY ↔ CASS

(SEE LD1)

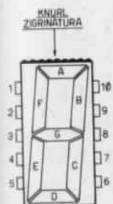
LD1 LD2 LD3 LD4 LD5 LD6 LD7 LD8

Q3 L203 Q2 L203



INTERCONNECTION CABLE BETWEEN NKZ80 KEY-BOARD AND J6 CLZ80/NC NANOCOMPUTER BOARD

CAVO DI CONNESSIONE FRA LA TASTIERA NKZ80 E LA SCHEDA CLZ80/NC (J6)



1 = COMMON CATHODE
2 = F
3 = G
4 = E
5 = D
6 = COMMON CATHODE
7 = DECIMAL POINT
8 = C
9 = B
10 = A

SGS-ATES

Keyboard Schematic Diagram
Schema Elettrico Tastiera

NKZ80
ENKZ8001V07
SHEET 1 OF 1
9-1-80

APPENDICE E

BIBLIOGRAFIA

1. *The Compact Edition of the Oxford English Dictionary*, Oxford Univ. Press. 1971.
2. Rudolf F. Graf, *Modern Dictionary of Electronics*, Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1977.
3. James Martin, *Telecommunications and the Computer*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.
4. Abraham Marcus and John D. Lenk, *Computers for Technicians*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1973.
5. Microdata Corporation, *Microprogramming Handbook*, Santa Ana, California, 1971.
6. J. Blukis e M. Baker, *Practical Digital Electronics*, Hewlett-Packard Company, Santa Clara, California, 1974.
7. Donald E. Lancaster, *TTL Cookbook*, Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1974.
8. H.V. Malmstadt, C.G. Enke, e S.R. Crouch, *Instrumentation for Scientists Series, Moduls 3. Digital and Analog Data Conversions*, W.A. Benjamin, Inc. Menlo Park, California, 1973-4.
9. H.V. Malmstadt e C.G. Enke, *Digital Electronics for Scientists*, W.A. Benjamin, Inc., New York, 1969.
10. J.D. Lenk, *Handbook of Logic Circuits*, Reston Publishing Company, Inc., Reston, Virginia, 1972.
11. A. James Diefenderfer, *Principles of Electronic Instrumentation*, W.B. Saunders Company, Philadelphia, Pennsylvania, 1972.
12. P.R. Rony e D.G. Larsen, *Logic & Memory Experiments Using TTL Integrated Circuits, Book 2*, Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1979.
13. Robert L. Morris e John R. Miller, Editors, *Designing with TTL Integrated Circuits*, McGraw-Hill Book Company, New York, 1971.
14. Charles J. Sippl, *Microcomputer Dictionary and Guide*, Matrix Publishers, Inc., Champaign, Illinois, 1976.
15. Donald Eadie, *Introduction to the Basic Computer*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
16. Texas Instruments Incorporated, *Microprocessor Handbook*, Dallas, Texas, 1975.

Per queste caratteristiche, il sistema NANOCOMPUTER è particolarmente adatto non solo all'apprendimento a scuola, sotto la guida di un insegnante, ma anche per chi voglia individualmente prepararsi a questa nuova professione.





NBZ80-B. Scheda base, miniterminale, contenitore-alimentatore, Nanobook 1, manuale tecnico.

È possibile, attraverso un kit di espansione, passare dalla versione NBZ80-B alla NBZ80-S. In tal modo ogni studente può scegliere, graduandolo nel tempo, il livello di apprendimento più consono alle proprie esigenze. L'NBZ80-S è a sua volta ulteriormente espandibile per consentire l'approfondimento di un linguaggio ad alto livello, il Basic, soprattutto nelle sue interazioni con l'hardware.

Il sistema NANOCOMPUTER: un sistema modulare. Il NANOCOMPUTER, studiato espressamente per impieghi didattici, riunisce in sé un'elevata rigorosità di concezione e un'estrema flessibilità, essenziali in un processo di apprendimento teorico e sperimentale al contempo. Nella sua versione più semplice, NBZ80-B, il NANOCOMPUTER permette anche allo studente senza conoscenze specifiche di impadronirsi delle tecniche di programmazione dei microprocessori.





NBZ80-HL. Come NBZ80-S con 16k byte di RAM, tastiera alfanumerica con interfaccia video, 8k ROM di Basic su scheda addizionale, libro Basic Programming Primer, monitor TV (opzionale).

Con la versione NBZ80-S lo studente viene introdotto anche nelle tecniche di interfacciamento di un microprocessore con il mondo esterno e nei problemi di interazione tra hardware e software.

Desidero ricevere gratuitamente maggiori informazioni su:
☐ sistema NANOCOMPUTER®
☐ corsi sullo Z80 con l'utilizzo del NANOCOMPUTER®

NB

NOME ______ COGNOME ______

INDIRIZZO ______

PROFESSIONE ______

Inviare a: SGS-ATES
Componenti Elettronici S p A
Via C. Olivetti 2-20041
Agrate Brianza, tel. (039) 65551



Quintino 29/C, 10121 Torino, Tel. (011) 531267 • DISTRIBUTORI: CID, Roma, Tel. (06) 6383979 • DE DO Electronic Fitting & C. S.a.s.,Tortoreto Lido (Te), Tel. (0861) 78134 • DISELCO S.p.A., Milano, Tel. (02) 3086141 • FANTON ELECTRONIC SYSTEM S.r.l., Padova, Tel. (049) 654487 • FANTON BOLOGNA S.r.l., Bologna, Tel. (051) 357300 • AGENTE ESCLUSIVO PER LA SCUOLA: GB PARAVIA & C. EDITORI, Torino, Tel. (011) 779166.

CARATTERISTICHE TECNICHE DEL NANOCOMPUTER NBZ80-S

Scheda base

FORMATO	Doppio europeo
MICROPROCESSORE	SGS-ATES Z80 CPU
CLOCK DI SISTEMA	2,4756 MHz
MEMORIA ROM	2K byte di Monitor NCZ80, espandibile sulla scheda fino a 8K byte (M2716 o M2316)
MEMORIA RAM	4K byte (M4027), espandibile sulla scheda fino a 16K byte (M4116).
I/O PARALLELO	2 Z80 PIO, due porte di 8 bit disponibili.
I/O SERIALE	Interfaccia seriale TTL, RS232C, anello di corrente 20mA. Interfaccia registratore a cassette audio RCZ80.
BUS DI SISTEMA	Interfaccia gamma-BUS standard.

Scheda per esperimenti

- ☐ Basetta per collegamenti senza saldature con 840 fori
- ☐ 8 interruttori con logica antirimbalzo
- ☐ 2 interruttori a molla di ritorno con logica antirimbalzo
- ☐ 8 LED e driver
- ☐ 1 LED di alimentazione presente

Miniterminale

- ☐ 8 cifre a 7 segmenti LED
- ☐ 14 LED
- ☐ 30 tasti
- ☐ 1 interruttore terminale seriale/registratore a cassette

Alimentatore

- ☐ 220V ±10%, 50Hz
- ☐ + 5V (± 5%), 1,5A (0,5A disponibili all'utente)
- ☐ − 5V (± 5%), 0,3A (0,1A disponibili all'utente)
- ☐ + 12V (± 5%), 0,4A (0,1A disponibili all'utente)
- ☐ − 12V (± 5%), 0,3A (0,1A disponibili all'utente)

Peso e dimensioni

- ☐ Kg. 3,4
- ☐ 35,7 cm. (largh.) X 40,0 cm. (lungh.) X 5,0 cm. (alt.)

Condizioni ambientali di lavoro

- ☐ Temperatura 0-50°C
- ☐ Umidità 0-90% (senza condensazione)

COMBINAZIONI, OPZIONI ED ACCESSORI

Estratto dal catalogo Nanocomputer Boards and Systems

Codice d'Ordinazione	Descrizione
NBZ80-B	Scheda base Nanocomputer con 2K ROM di Monitor, miniterminale e contenitore/alimentatore. Comprende Manuale tecnico, Nanobook 1.
NBZ80-S	Come NBZ80-B più scheda per esperimenti inserita nel contenitore/alimentatore, kit di fili K1Z80 e Nanobook 3.
NBZ80-HL	Come NBZ80-S più interfaccia monitor TV, tastiera alfanumerica, 8K ROM di BASIC, guida introduttiva al BASIC. Monitor TV (TVZ80) non incluso.
UPZ80-BS	Parti per espandere NBZ80-B in NBZ80-S e documentazione.
UPZ80-HL	Parti per espandere NBZ80-S in NBZ80-HL e documentazione.
KNZ80	Kit per convertire la scheda base del Nanocomputer in una scheda CLZ80-4/2.
RCZ80	Registratore a cassette audio.
TVZ80	Monitor televisivo utilizzabile con il Nanocomputer NBZ80-HL.
K1Z80	Fili già tagliati e spellati per esperimenti.
K2Z80	Kit di componenti per gli esperimenti descritti nel Nanobook 3.
NE-Z	ROM e documentazione per i programmi descritti nel Nanobook 3.
DAZ80NANOBK1/1	Nanobook 1. Tecniche di programmazione dello Z80.
DAZ80NANOBK3/1	Nanobook 3. Tecniche di interfacciamento dello Z80.
DAZ80SIST/1	Set di istruzioni CPU Z80.
DAZ80CPUMT/1	Manuale tecnico su CPU, PIO e CTC della famiglia Z80.
PBZTEMPLATE	Mascherina per diagrammi di flusso e pieghevole istruzioni Z80.
PBBLOCK1-Z	Block notes per scrivere programmi.
PBFILE-Z	Raccoglitore per documentazione.

SGS-ATES COMPONENTI ELETTRONICI SpA Via C. Olivetti 2, 20041 Agrate Brianza, Tel. (039) 65551 ●DIREZIONE COMMERCIALE ITALIA Via Correggio 1/3, 20149 Milano, Tel. (02) 4695651 ●UFFICI VENDITA Via G. Del Pian dei Carpini 96/1, 50127 Firenze, Tel. (055) 4377763 ●Via Correggio 1/3, 20149 Milano, Tel. (02) 4695651 ● Piazza Gondar 11, 00199 Roma, Tel. (06) 8392848 ●Corso G. Ferraris 26, 10121 Torino, Tel. (011) 531167 ● PUNTI DI VENDITA Via Larga, ang. Via Brolo/Via Verziere, 20122 Milano, Tel. (02) 8690047 ● Via S. Quintino 29/C, 10121 Torino, Tel. (011) 532167.



L. 18.000

Z-80 = Trade Mark della Zilog Inc.
Nanobook = Marchio Registrato della SGS-ATES S.p.A.



La dotto.ssa *Elizabeth A. Nichols* è consulente di sistemi presso la CENTEC Corporation Reston, Virginia, specializzata nelle applicazioni della microelettronica ai problemi di tipo energetico ed ambientale. La Nichols ha conseguito il P.h.D. in matematica alla Duke University nel 1974.



Il dott. *Joseph C. Nichols* è consulente di sistemi alla Network Analysis Corporation, Washington, specializzata nell'analisi e nella progettazione di reti di comunicazione e sistemi distribuiti. Ha conseguito il P.h.D. in matematica alla Duke University nel 1970.



Il dott. *Peter R. Rony* è professore al Dipartimento di Ingegneria Chimica al Virginia Polytechnic Institute & State University. Rivolge molta attenzione all'elettronica digitale ed ai microcomputer da quando essi occupano un ruolo sempre più considerevole nel campo del controllo di processo. E' coautore di molti libri della Blacksburg Continuing Education Series TM e di articoli mensili sull'interfacciamento dei microcomputer che appaiono sulle riviste american Laboratory, Computer Design, Ham Radio Magazine, la rivista tedesca Electroniker, la rivista italiana Elettronica Oggi e altre riviste americane ed europee.